Package ‘FRESA.CAD’

November 27, 2018

Type     Package
Title    Feature Selection Algorithms for Computer Aided Diagnosis
Version  3.1.0
Date     2018-10-30
Author   Jose Gerardo Tamez-Pena, Antonio Martinez-Torteya and Israel Alanis
Maintainer Jose Gerardo Tamez-Pena <jose.tamezpena@tec.mx>
Description Contains a set of utilities for building and testing statistical models (linear, logistic, ordinal or COX) for Computer Aided Diagnosis/Prognosis applications. Utilities include data adjustment, univariate analysis, model building, model-validation, longitudinal analysis, reporting and visualization.
License   LGPL (>= 2)
Depends  Rcpp (>= 0.10.0), stringr, miscTools, Hmisc, pROC
LinkingTo Rcpp, RcppArmadillo
Suggests nlme, rpart, gplots, RColorBrewer, class, cvTools, glmnet, randomForest, survival, e1071, MASS, naivebayes, mRMRe, epiR, DescTools
NeedsCompilation yes
Repository CRAN
Date/Publication 2018-11-27 08:50:03 UTC

R topics documented:

FRESA.CAD-package ............................................. 3
backVarElimination_Bin ....................................... 8
backVarElimination_Res ...................................... 10
baggedModel .................................................. 11
barPlotCIError ............................................... 13
benchmarking ............................................... 14
bootstrapValidation_Bin .................................. 18
bootstrapValidation_Res .................................. 21
bootstrapVarElimination_Bin ......................... 23
bootstrapVarElimination_Res ....................... 25
R topics documented:

- BSWiMS.model .............................................. 26
- cancerVarNames .......................................... 31
- crossValidationFeatureSelection_Bin .................. 32
- crossValidationFeatureSelection_Res .................. 37
- CVsignature .................................................. 41
- EmpiricalSurvDiff ......................................... 42
- ensemblePredict ............................................ 44
- featureAdjustment ......................................... 45
- FilterUnivariate ........................................... 46
- ForwardSelection.Model.Bin ............................. 48
- ForwardSelection.Model.Res ............................. 50
- FRESA.Model ................................................ 52
- FRESAScale .................................................. 56
- getKNNpredictionFromFormula ......................... 57
- getSignature ............................................... 58
- getVar.Bin ................................................... 60
- getVar.Res ................................................... 61
- heatMaps ...................................................... 63
- improvedResiduals ......................................... 65
- KNN_method .................................................. 66
- LASSO .......................................................... 67
- listTopCorrelatedVariables ............................... 68
- LM_RIDGE_MIN .............................................. 69
- modelFitting ................................................. 70
- mRMR.classic_FRESA ....................................... 71
- NAIVE_BAYES ............................................... 71
- nearestNeighborImpute .................................... 72
- plot.bootstrapValidation_Bin ......................... 73
- plot.bootstrapValidation_Res ......................... 74
- plot.FRESA_benchmark .................................... 75
- plotPredict.FRESA .......................................... 76
- predict.fitFRESA ........................................... 77
- predict.FRESA_KNN ......................................... 78
- predict.FRESA_signature ................................. 78
- predict.FRESA_LASSO ...................................... 79
- predict.FRESA_NAIVEBAYES .............................. 80
- predict.FRESA_RIDGE ...................................... 80
- predictionStats ............................................. 81
- randomCV ..................................................... 82
- rankInverseNormalDataFrame ............................. 85
- reportEquivalentVariables ............................... 87
- residualForFRESA .......................................... 88
- signatureDistance ......................................... 89
- summary_bootstrapValidation_Bin ...................... 90
- summary.fitFRESA .......................................... 91
- summaryReport ............................................. 92
- timeSerieAnalysis ......................................... 93
- uniRankVar .................................................. 94
Description

Contains a set of utilities for building and testing formula-based models for Computer Aided Diagnosis/prognosis applications via feature selection. Bootstrapped Stage Wise Model Selection (B:SWiMS) controls the false selection (FS) for linear, logistic, or Cox proportional hazards regression models. Utilities include functions for: univariate/longitudinal analysis, data conditioning (i.e. covariate adjustment and normalization), model validation and visualization.

Details

Package: FRESA.CAD
Type: Package
Version: 3.1.0
Date: 2018-10-30
License: LGPL (>= 2)

Purpose: The design of diagnostic or prognostic multivariate models via the selection of significantly discriminant features. The models are selected via the bootstrapped step-wise selection of model features that offer a significant improvement in subject classification/error. The false selection control is achieved by train-test partitions, where train sets are used to select variables and test sets used to evaluate model performance. Variables that do not improve subject classification/error on the blind test are not included in the models. The main function of this package is the selection and cross-validation of diagnostic/prognostic linear, logistic, or Cox proportional hazards regression model constructed from a large set of candidate features. The variable selection may start by conditioning all variables via a covariate-adjustment and a z-inverse-rank-transformation. In order to integrate features with partial discriminant power, the package can be used to categorize the continuous variables and rank their discriminant power. Once ranked, each feature is bootstrap-tested in a multivariate model, and its blind performance is evaluated. Variables with a statistical significant improvement in classification/error are stored and finally inserted into the final model according to their relative store frequency. A cross-validation procedure may be used to diagnose the amount of model shrinkage produced by the selection scheme.

Author(s)

Jose Gerardo Tamez-Pena, Antonio Martinez-Torteya and Israel Alanis Maintainer: <jose.tamezpena@tec.mx>
References


Examples

```r
## Not run:
### Fresa Package Examples ####
library("epir")
library("FRESA.CAD")
library(network)
library(GGally)
library("e1071")

# Start the graphics device driver to save all plots in a pdf format

# Get the stage C prostate cancer data from the rpart package
data(stagec, package = "rpart")
options(na.action = 'na.pass')
dataCancer <- cbind(pgstat = stagec$pgstat,
pctime = stagec$pctime,
   as.data.frame(model.matrix(Surv(pctime, pgstat) ~ ., stagec))[-1])

#Impute missing values
dataCancerImputed <- nearestNeighborImpute(dataCancer)

# Remove the incomplete cases
dataCancer <- dataCancer[complete.cases(dataCancer),]

# Load a pre-established data frame with the names and descriptions of all variables
data(cancerVarNames)
# the Heat Map
hm <- heatMaps(cancerVarNames, varRank=NULL, Outcome="pgstat",
data=dataCancer, title="Heat Map", hCluster=FALSE,
prediction=NULL, Scale=TRUE,
theFiveColors=c("blue", "cyan", "black", "yellow", "red"),
outcomeColors =
   c("blue", "lightgreen", "yellow", "orangered", "red"),
transpose=FALSE, cexRow=0.5, cexCol=0.8, srtCol=35)

# The univariate analysis
UniRankFeaturesRaw <- univariateRankVariables(variableList = cancerVarNames,
   formula = "pgstat ~ 1+pctime",
   Outcome = "pgstat",
   data = dataCancer,
categorizationType = "Raw",
   stagec)
```
type = "LOGIT",
rankingTest = "zIDI",
description = "Description",
uniType = "Binary"
)

print(UniRankFeaturesRaw)

# A simple BSIWMS Model
BSWiMSModel <- BSWiMS.model(formula = Surv(pgttime, pgstat) ~ 1, dataCancerImputed)

# The Log-Rank Analysis using survdiff
lrsurvdiff <- survdiff(Surv(pgttime, pgstat) ~ BSWiMSModel$BSWiMS.model$back.model$linear.predictors > 0,
data = dataCancerImputed)

# The Log-Rank Analysis EmpiricalSurvDiff and permutations of the null Chi distribution
lrp <- EmpiricalSurvDiff(dataCancerImputed$pgtime, dataCancerImputed$pgstat,
BSWiMSModel$BSWiMS.model$back.model$linear.predictors > 0,
type = "Chi", plots = TRUE, samples = 10000)

# The Log-Rank Analysis EmpiricalSurvDiff and permutations of the null SLR distribution
lrp <- EmpiricalSurvDiff(dataCancerImputed$pgtime, dataCancerImputed$pgstat,
BSWiMSModel$BSWiMS.model$back.model$linear.predictors > 0,
type = "SLR", plots = TRUE, samples = 10000)

# The Log-Rank Analysis EmpiricalSurvDiff and bootstrapping the SLR distribution
lrp <- EmpiricalSurvDiff(dataCancerImputed$pgtime, dataCancerImputed$pgstat,
BSWiMSModel$BSWiMS.model$back.model$linear.predictors > 0,
computeDist = TRUE, plots = TRUE)

# The performance of the final model using the summary function
sm <- summary(BSWiMSModel$BSWiMS.model$back.model)
print(sm$coefficients)
pv <- plot(sm$bootstrap)

# The equivalent model
eq <- reportEquivalentVariables(BSWiMSModel$BSWiMS.model$back.model, data = dataCancer,
variableList = cancerVarNames, Outcome = "pgstat",
timeOutcome = "pgtime",
type = "COX");

print(eq$equivalentMatrix)

# The list of all models of the bootstrap forward selection
print(BSWiMSModel$forward.selection.list)

# With FRESA.CAD we can do a leave-one-out using the list of models
pm <- ensemblePredict(BSWiMSModel$forward.selection.list,
dataCancer, predictType = "linear", type = "LOGIT", Outcome = "pgstat")

# Plotting the ROC with 95
pm <- plotModels.ROC(cbind(dataCancer$pgstat, 
  pm$ensemblePredict),main="(LOO Forward Selection Median Predict")

#The plotModels.ROC provides the diagnosis confusion matrix.
summary(epi.tests(pm$predictionTable))

#FRESA.CAD can be used to create a bagged model using the forward selection formulas
bagging <- baggedModel(BSWiMSModel$forward.selection.list,dataCancer,useFreq=32)
pm <- predict(bagging$bagged.model)
pm <- plotModels.ROC(cbind(dataCancer$pgstat,pm),main="(Bagged")

#Let's check the performance of the model
sm <- summary(bagging$bagged.model)
print(sm$coefficients)

#Using bootstrapping object I can check the Jaccard Index
print(bagging$Jaccard.SM)

#Ploting the evolution of the coefficient value
plot(bagging$coefEvolution$grade,main="Evolution of grade")

gplots::heatmap.2(bagging$formulaNetwork,trace="none",
  mar=c(10,10),main="eB: SWIMS Formula Network")
barplot(bagging$frequencyTable,las = 2, cex.axis=1.0,
  cex.names=0.75,main="Feature Frequency")
n <- network::network(bagging$formulaNetwork, directed = FALSE,
  ignore.eval = FALSE, names.eval = "weights")
ggnet(n, label = TRUE, size = "degree", size.cut = 3, size.min = 1,
  mode = "circle", edge.label = "weights", edge.label.size=4)

# Get a Cox proportional hazards model using:
# The default parameters
mdCOXs <- FRESA.Model(formula = Surv(pertime, pgstat) ~ 1, data = dataCancer)
sm <- summary(mdCOXs$BSWIMS.model)
print(sm$coefficients)

# The model with significant improvement in the residual error
mdCOXs <- FRESA.Model(formula = Surv(pertime, pgstat) ~ 1, 
  data = dataCancer, OptType = "Residual")
sm <- summary(mdCOXs$BSWIMS.model)
print(sm$coefficients)

# Get a Cox proportional hazards model using second order models:
mdCOX <- FRESA.Model(formula = Surv(pertime, pgstat) ~ 1, 
  data = dataCancer, categorizationType="RawRaw")
sm <- summary(mdCOX$BSWIMS.model)
print(sm$coefficients)
namesc <- names(mdCOX$BSWiMS.models$coefficients)[-1]
hm <- heatMaps(mdCOX$univariateAnalysis[namesc,], varRank=NULL,
  Outcome = "pgstat", data = dataCancer,
  title = "Heat Map", hCluster = FALSE, prediction = NULL, Scale = TRUE,
  theFiveColors = c("blue", "cyan", "black", "yellow", "red"),
  outcomeColors = c("blue", "lightgreen", "yellow", "orangered", "red"),
  transpose = FALSE, cexRow = 0.50, cexCol = 0.80, srtCol = 35)
# The LOO estimation
pm <- ensemblePredict(mdCOX$BSWiMS.models$formula.list, dataCancer,
  predictType = "linear", type = "LOGIT", Outcome = "pgstat")
pm <- plotModels.ROC(cbind(dataCancer$pgstat, pm$ensemblePredict), main = "(LOO Median Predict")
# Let us check the diagnosis performance
summary(epi.tests(pm$predictionTable))

# Get a Logistic model using FRESA.Model
# - The default parameters
dataCancer2 <- dataCancer
dataCancer2$pgrtime <- NULL
mdLOGIT <- FRESA.Model(formula = pgstat ~ 1, data = dataCancer2)
if (!is.null(mdLOGIT$bootstrappedModel)) pv <- plot(mdLOGIT$bootstrappedModel)
sm <- summary(mdLOGIT$BSWiMS.model)
print(sm$coefficients)

## FRESA.Model with Cross Validation and Recursive Partitioning and Regression Trees
md <- FRESA.Model(formula = Surv(pgtime, pgstat) ~ 1, data = dataCancer,
  CVfolds = 10, repeats = 5, equivalent = TRUE, usrFitFun = rpart:::rpart)
colnames(md$cvObject$Models$testPrediction)

pm <- plotModels.ROC(md$cvObject$Models$testPredictions, theCVfolds = 10, main = "CV LASSO", cex = 0.90)
pm <- plotModels.ROC(md$cvObject$Models$testPrediction, theCVfolds = 10, main = "KNN", cex = 0.90)

pm <- plotModels.ROC(md$cvObject$Models$testPrediction, theCVfolds = 10,
  predictor = "Prediction", main = "BSWiMS Bagging", cex = 0.90)

pm <- plotModels.ROC(md$cvObject$Models$testPrediction, theCVfolds = 10,
  predictor = "Ensemble.B.SWiMS",
  main = "Forward Selection Median Ensemble", cex = 0.90)

pm <- plotModels.ROC(md$cvObject$Models$testPrediction, theCVfolds = 10,
  predictor = "Ensemble.Forward", main = "Forward Selection Bagging", cex = 0.90)

pm <- plotModels.ROC(md$cvObject$Models$testPrediction, theCVfolds = 10,
  predictor = "eB.SWiMS", main = "Equivalent Model", cex = 0.90)

pm <- plotModels.ROC(md$cvObject$Models$testPrediction, theCVfolds = 10,
  predictor = "Forward.Selection.Bagged", main = "The Forward Bagging", cex = 0.90)

pm <- plotModels.ROC(md$cvObject$Models$testPrediction, theCVfolds = 20,
  predictor = "usrFitFunction",
  main = "Recursive Partitioning and Regression Trees", cex = 0.90)

pm <- plotModels.ROC(md$cvObject$Models$testPrediction, theCVfolds = 20,
  predictor = "usrFitFunctionSel",
  main = "Recursive Partitioning and Regression Trees with FS", cex = 0.90)
## IDI/NRI-based backwards variable elimination

**Description**

This function removes model terms that do not significantly affect the integrated discrimination improvement (IDI) or the net reclassification improvement (NRI) of the model.

**Usage**

```r
backVarElimination_Bin(object, 
  pvalue = 0.05, 
  Outcome = "Class", 
  data, 
  startOffset = 0, 
  type = c("LOGIT", "LM", "COX"), 
  selectionType = c("zIDI", "zNRI")
)
```
Arguments

- **object**: An object of class `lm`, `glm`, or `coxph` containing the model to be analyzed.
- **pvalue**: The maximum $p$-value, associated to either IDI or NRI, allowed for a term in the model.
- **Outcome**: The name of the column in `data` that stores the variable to be predicted by the model.
- **data**: A data frame where all variables are stored in different columns.
- **startOffset**: Only terms whose position in the model is larger than the `startOffset` are candidates to be removed.
- **type**: Fit type: Logistic ("LOGIT"), linear ("LM"), or Cox proportional hazards ("COX").
- **selectionType**: The type of index to be evaluated by the `improveProb` function (`hmisc` package): $z$-score of IDI or of NRI.

Details

For each model term $x_i$, the IDI or NRI is computed for the Full model and the reduced model (where the term $x_i$ removed). The term whose removal results in the smallest drop in improvement is selected. The hypothesis: the term adds classification improvement is tested by checking the `pvalue` of improvement. If $p(IDIorNRI) > pvalue$, then the term is removed. In other words, only model terms that significantly aid in subject classification are kept. The procedure is repeated until no term fulfills the removal criterion.

Value

- **back.model**: An object of the same class as `object` containing the reduced model.
- **loops**: The number of loops it took for the model to stabilize.
- **reclas.info**: A list with the NRI and IDI statistics of the reduced model, as given by the `getVar.Bin` function.
- **back.formula**: An object of class `formula` with the formula used to fit the reduced model.
- **lastRemoved**: The name of the last term that was removed (-1 if all terms were removed).
- **at.opt.model**: The model before the BH procedure.
- **beforeFSC.formula**: The string formula of the model before the BH procedure.

Author(s)

Jose G. Tamez-Pena and Antonio Martinez-Torteya

References


See Also

`backVarElimination_Res`, `bootstrapVarElimination_Bin`, `bootstrapVarElimination_Res`
Description

This function removes model terms that do not significantly improve the "net residual" (NeRI).

Usage

```r
backVarElimination_Res(object, pvalue = 0.05, Outcome = "Class", data, startOffset = 0, type = c("LOGIT", "LM", "COX"), testType = c("Binomial", "Wilcoxon", "tStudent", "Ftest"), setIntersect = 1)
```

Arguments

- **object**: An object of class `lm`, `glm`, or `coxph` containing the model to be analyzed
- **pvalue**: The maximum p-value, associated to the NeRI, allowed for a term in the model
- **Outcome**: The name of the column in data that stores the variable to be predicted by the model
- **data**: A data frame where all variables are stored in different columns
- **startOffset**: Only terms whose position in the model is larger than the startOffset are candidates to be removed
- **type**: Fit type: Logistic ("LOGIT"), linear ("LM"), or Cox proportional hazards ("COX")
- **testType**: Type of non-parametric test to be evaluated by the improvedResiduals function: Binomial test ("Binomial"), Wilcoxon rank-sum test ("Wilcoxon"), Student’s t-test ("tStudent"), or F-test ("Ftest")
- **setIntersect**: The intersect of the model (To force a zero intersect, set this value to 0)

Details

For each model term $x_i$, the residuals are computed for the Full model and the reduced model (where the term $x_i$ removed). The term whose removal results in the smallest drop in residuals improvement is selected. The hypothesis: the term improves residuals is tested by checking the pvalue of improvement. If $p(\text{residuals better than reduced residuals}) > pvalue$, then the term is removed. In other words, only model terms that significantly aid in improving residuals are kept. The procedure is repeated until no term fulfills the removal criterion. The p-values of improvement can be computed via a sign-test (Binomial) a paired Wilcoxon test, paired t-test or f-test. The first three tests compare the absolute values of the residuals, while the f-test test if the variance of the residuals is improved significantly.
Value

- back.model: An object of the same class as object containing the reduced model
- loops: The number of loops it took for the model to stabilize
- reclass.info: A list with the NeRI statistics of the reduced model, as given by the getVar.Res function
- back.formula: An object of class formula with the formula used to fit the reduced model
- lastRemoved: The name of the last term that was removed (-1 if all terms were removed)
- at.opt.model: the model with before the FSR procedure.
- beforeFSC.formula: the string formula of the the FSR procedure

Author(s)

Jose G. Tamez-Pena and Antonio Martinez-Torteya

See Also

backVarElimination_Bin, bootstrapVarElimination_Bin bootstrapVarElimination_Res

Description

This function will take the frequency-ranked of variables and the list of models to create a single bagged model

Usage

```r
baggedModel(modelFormulas, data, type=c("LM","LOGIT","COX"),
Outcome=NULL, timeOutcome=NULL, frequencyThreshold=0.025,
univariate=NULL, useFreq=TRUE, n_bootstrap=1)
```
Arguments

**modelFormulas**  The name of the column in `data` that stores the variable to be predicted by the model

**data**  A data frame with two columns. The first one must have the names of the candidate variables and the other one the description of such variables

**type**  Fit type: Logistic ("LOGIT"), linear ("LM"), or Cox proportional hazards ("COX")

**Outcome**  The name of the column in `data` that stores the time to outcome

**timeOutcome**  The name of the column in `data` that stores the time to event (needed only for a Cox proportional hazards regression model fitting)

**frequencyThreshold**  set the frequency the threshold of the frequency of features to be included in the model

**univariate**  The FFRESA.CAD univariate analysis matrix

**useFreq**  Use the feature frequency to order the formula terms. If set to a positive value is the number of estimation loops

**n_bootstrap**  if greater than 1, defines the number of bootstraps samples to be used

Value

**baggedModel**  the bagged model

**formula**  the formula of the model

**frequencyTable**  the table of variables ranked by their model frequency

**faverageSize**  the average size of the models

**formulaNetwork**  The matrix of interaction between formulas

**Jaccard.SM**  The Jaccard Stability Measure of the formulas

**coefEvolution**  The evolution of the coefficients

**avgZvalues**  The average Z value of each coefficient

**featureLocation**  The average location of the feature in the formulas

Author(s)

Jose G. Tamez-Pena

See Also

`ensemblePredict`
\textit{barPlotCiError} \hspace{1cm} \textit{Bar plot with error bars}

\textbf{Description}

Ranked Plot a set of measurements with error bars or confidence intervals (CI)

\textbf{Usage}

\begin{verbatim}
barPlotCiError(ciTable, metricname, thesets, themethod, main, angle = 0, offsets = c(0.1,0.1), scoreDirection = ">", ho=NULL, ...)
\end{verbatim}

\textbf{Arguments}

- \texttt{ciTable} \hspace{1cm} A matrix with three columns: the value, the low CI value and the high CI value
- \texttt{metricname} \hspace{1cm} The name of the plotted values
- \texttt{thesets} \hspace{1cm} A character vector with the names of the sets
- \texttt{themethod} \hspace{1cm} A character vector with the names of the methods
- \texttt{main} \hspace{1cm} The plot title
- \texttt{angle} \hspace{1cm} The angle of the x labels
- \texttt{offsets} \hspace{1cm} The offset of the x-labels
- \texttt{scoreDirection} \hspace{1cm} Indicates how to aggregate the supMethod score and the ingMethod score.
- \texttt{ho} \hspace{1cm} the null hypothesis
- \texttt{...} \hspace{1cm} Extra parameters passed to the barplot function

\textbf{Value}

- \texttt{barplot} \hspace{1cm} the x-location of the bars
- \texttt{ciTable} \hspace{1cm} the ordered matrix with the 95 CI
- \texttt{barmatrix} \hspace{1cm} the mean values of the bars
- \texttt{supMethod} \hspace{1cm} A superiority score equal to the numbers of methods that were inferior
- \texttt{infMethod} \hspace{1cm} A inferiority score equal to the number of methods that were superior
- \texttt{interMethodScore} \hspace{1cm} the sum of supMethod and infMethod defined by the score direction.
**Author(s)**

Jose G. Tamez-Pena

---

**Compare performance of different model fitting/filtering algorithms**

**Description**

Evaluates a data set with a set of fitting/filtering methods and returns the observed cross-validation performance.

**Usage**

```r
BinaryBenchmark(theData = NULL, theOutcome = "Class", reps = 100, trainFraction = 0.5,
referenceCV = NULL, referenceName = "Reference",
referenceFilterName="Reference")
RegressionBenchmark(theData = NULL, theOutcome = "Class", reps = 100, trainFraction = 0.5,
referenceCV = NULL, referenceName = "Reference",
referenceFilterName="Reference")
OrdinalBenchmark(theData = NULL, theOutcome = "Class", reps = 100, trainFraction = 0.5,
referenceCV = NULL, referenceName = "Reference",
referenceFilterName="Reference")
```

**Arguments**

- `theData` The data frame
- `theOutcome` The outcome feature
- `reps` The number of times that the random cross-validation will be performed
- `trainFraction` The fraction of the data used for training.
- `referenceCV` A user supplied random cross-validation object to be benchmarked
- `referenceName` The name of the reference classifier to be used in the reporting tables
- `referenceFilterName` The name of the reference filter to be used in the reporting tables

**Details**

The benchmark functions provide the performance of different classification algorithms (BinaryBenchmark), registration algorithms (RegressionBenchmark) or ordinal regression algorithms (OrdinalBenchmark). The evaluation method is based on applying the random cross-validation method (`randomCV`) that randomly splits the data into train and test sets. The user can provide a Cross validated object that will define the train-test partitions.
The BinaryBenchmark compares: BSWiMS, Random Forest, RPART, LASSO, SVM/mRMR, KNN and the ensemble of them in their ability to correctly classify the test data. Furthermore, it evaluates the ability of the following feature selection algorithms: BSWiMS or ReferenceCV, LASSO, RPART, RF/BSWiMS, IDI, NRI, t-test, Wilcoxon, Kendall, and mRMR in their ability to select the best set of features for the following classification methods: SVM, KNN, Naive Bayes, Random Forest Nearest Centroid (NC) with root sum square (RSS), and NC with Spearman correlation.

The RegressionBenchmark compares: BSWiMS, Random Forest, RPART, LASSO, SVM/mRMR and the ensemble of them in their ability to correctly predict the test data. Furthermore, it evaluates the ability of the following feature selection algorithms: BSWiMS or ReferenceCV, LASSO, RPART, RF/BSWiMS, F-Test, W-Test, Pearson Kendall, and mRMR in their ability to select the best set of features for the following regression methods: Linear Regression, Robust Regression, Ridge Regression, LASSO, SVM, and Random Forest.

The OrdinalBenchmark compares: BSWiMS, Random Forest, RPART, LASSO, KNN, SVM and the ensemble of them in their ability to correctly predict the test data. Furthermore, it evaluates the ability of the following feature selection algorithms: BSWiMS or ReferenceCV, LASSO, RPART, RF/BSWiMS, F-Test, Kendall, and mRMR in their ability to select the best set of features for the following regression methods: Ordinal, KNN, SVM, Random Forest, and Naive Bayes.

Value

- errorTable: the matrix of the balanced error with the 95 CI
- accTable: the matrix of the classification accuracy with the 95 CI
- aucTable: the matrix of the ROC AUC with the 95 CI
- senTable: the matrix of the sensitivity with the 95 CI
- speTable: the matrix of the specificity with the 95 CI
- errorTable_filter: the matrix of the balanced error with the 95 CI for filter methods
- accTable_filter: the matrix of the classification accuracy with the 95 CI for filter methods
- senTable_filter: the matrix of the classification sensitivity with the 95 CI for filter methods
- speTable_filter: the matrix of the classification specificity with the 95 CI for filter methods
- aucTable_filter: the matrix of the ROC AUC with the 95 CI for filter methods
- CorTable: the matrix of the Pearson correlation with the 95 CI
- RMSETable: the matrix of the root mean square error (RMSE) with the 95 CI
- BiasTable: the matrix of the prediction bias with the 95 CI
- CorTable_filter: the matrix of the Pearson correlation with the 95 CI for filter methods
- RMSETable_filter: the matrix of the root mean square error (RMSE) with the 95 CI for filter methods
- BiasTable_filter: the matrix of the prediction bias with the 95 CI for filter methods
BMAETable the matrix of the balanced mean absolute error (MEA) with the 95 CI for filter methods
KappaTable the matrix of the Kappa value with the 95 CI
BiasTable the matrix of the prediction Bias with the 95 CI
KendallTable the matrix of the Kendall correlation with the 95 CI
MAETable_filter the matrix of the mean absolute error (MEA) with the 95 CI for filter methods
KappaTable_filter the matrix of the Kappa value with the 95 CI for filter methods
BiasTable_filter the matrix of the prediction Bias with the 95 CI for filter methods
KendallTable_filter the matrix of the Kendall correlation with the 95 CI for filter methods
times The average CPU time used by the method
jaccard_filter The average Jaccard Index of the feature selection methods
TheCVEvaluations The output of the randomCV (randomCV) evaluations of the different methods
testPredictions A matrix with all the test predictions
featureSelectionFrequency The frequency of feature selection
cpuElapsedTimes The mean elapsed times

data(stagec, package = "rpart")

data <- stagec, package = "rpart")

# Prepare the data. Create a model matrix without the event time
stagec$ptime <- NULL
stagec$eet <- as.factor(stagec$eet)
options(na.action = 'na.pass')
stagec_mat <- cbind(pgstat = stagec$pgstat,
as.data.frame(model.matrix(pgstat ~ ., stagec))[-1])

# Impute the missing data
dataCancerImputed <- nearestNeighborImpute(stagec_mat)
dataCancerImputed[,1:ncol(dataCancerImputed)] <- sapply(dataCancerImputed,as.numeric)

# Cross validating a LDA classifier.
# 80
cv <- randomCV(dataCancerImputed,"pgstat",MASS::lda,trainFraction = 0.8,
repetitions = 10,featureSelectionFunction = univariate_tstudent,
featureSelection.control = list(limit = 0.5,thr = 0.975));

# Compare the LDA classifier with other methods
cp <- BinaryBenchmark(referenceCV = cv,referenceName = "LDA",
referenceFilterName="t.Student")
pl <- plot(cp,prefix = "StageC: ");

# Default Benchmark classifiers method (BWiMS) and filter methods.
# 80
cp <- BinaryBenchmark(theData = dataCancerImputed,
theOutcome = "pgstat", reps = 10, fraction = 0.8)

# plot the Cross Validation Metrics
pl <- plot(cp,prefix = "Stagec:");

# Shut down the graphics device driver
dev.off()

### Regression Example #####
# Start the graphics device driver to save all plots in a pdf format
dpdf(file = "RegressionExample.pdf",width=8, height=6)

# Get the body fat data from the TH package
data("bodyfat", package = "TH.data")

# Benchmark regression methods and filter methods.
# 80
cp <- RegressionBenchmark(theData = bodyfat,
theOutcome = "DEXfat", reps = 10, fraction = 0.8)

# plot the Cross Validation Metrics
pl <- plot(cp,prefix = "Body Fat:");
# Shut down the graphics device driver
dev.off()

### Ordinal Regression Example #####
# Start the graphics device driver to save all plots in a pdf format
dpdf(file = "OrdinalRegressionExample.pdf",width=8, height=6)
# Get the GBSG2 data
data("GBSG2", package = "TH.data")

# Prepare the model frame for benchmarking
GBSG2$time <- NULL;
GBSG2$cens <- NULL;
GBSG2_mat <- cbind(tgrade = as.numeric(GBSG2$tgrade),
as.data.frame(model.matrix(tgrade~., GBSG2))[-1])

# Benchmark regression methods and filter methods.
#30
 cp <- OrdinalBenchmark(theData = GBSG2_mat,
    theOutcome = "tgrade", reps = 10, fraction = 0.3)

# plot the Cross Validation Metrics
pl <- plot(cp,prefix = "GBSG.");

# Shut down the graphics device driver
dev.off()

## End(Not run)

---

**Bootstrap validation of binary classification models**

**Description**

This function bootstraps the model \( n \) times to estimate for each variable the empirical distribution of model coefficients, area under ROC curve (AUC), integrated discrimination improvement (IDI) and net reclassification improvement (NRI). At each bootstrap the non-observed data is predicted by the trained model, and statistics of the test prediction are stored and reported. The method keeps track of predictions and plots the bootstrap-validated ROC. It may plots the blind test accuracy, sensitivity, and specificity, contrasted with the bootstrapped trained distributions.

**Usage**

```r
bootstrapValidation_Bin(fraction = 1,
    loops = 200,
    model.formula, Outcome, data,
    type = c("LM", "LOGIT", "COX"),
    plots = FALSE,
    best.model.formula=NULL)
```
Arguments

- **fraction**: The fraction of data (sampled with replacement) to be used as train
- **loops**: The number of bootstrap loops
- **model.formula**: An object of class `formula` with the formula to be used
- **Outcome**: The name of the column in `data` that stores the variable to be predicted by the model
- **data**: A data frame where all variables are stored in different columns
- **type**: Fit type: Logistic ("LOGIT"), linear ("LM"), or Cox proportional hazards ("COX")
- **plots**: Logical. If TRUE, density distribution plots are displayed
- **best.model.formula**: An object of class `formula` with the formula to be used for the best model

Details

The bootstrap validation will estimate the confidence interval of the model coefficients and the NRI and IDI. The non-sampled values will be used to estimate the blind accuracy, sensitivity, and specificity. A plot to monitor the evolution of the bootstrap procedure will be displayed if `plots` is set to TRUE. The plot shows the train and blind test ROC. The density distribution of the train accuracy, sensitivity, and specificity are also shown, with the blind test results drawn along the y-axis.

Value

- **data**: The data frame used to bootstrap and validate the model
- **outcome**: A vector with the predictions made by the model
- **blind.accuracy**: The accuracy of the model in the blind test set
- **blind.sensitivity**: The sensitivity of the model in the blind test set
- **blind.specificity**: The specificity of the model in the blind test set
- **train.ROCAUC**: A vector with the AUC in the bootstrap train sets
- **blind.ROCAUC**: An object of class roc containing the AUC in the bootstrap blind test set
- **boot.ROCAUC**: An object of class roc containing the AUC using the mean of the bootstrapped coefficients
- **fraction**: The fraction of data that was sampled with replacement
- **loops**: The number of loops it took for the model to stabilize
- **base.Accuracy**: The accuracy of the original model
- **base.sensitivity**: The sensitivity of the original model
- **base.specificity**: The specificity of the original model
- **accuracy**: A vector with the accuracies in the bootstrap test sets
sensitivities   A vector with the sensitivities in the bootstrap test sets
specificities  A vector with the specificities in the bootstrap test sets
train.accuracy A vector with the accuracies in the bootstrap train sets
train.sensitivity
                    A vector with the sensitivities in the bootstrap train sets
train.specificity A vector with the specificities in the bootstrap train sets
s.coef          A vector with the coefficients in the bootstrap train sets
boot.model      An object of class lm, glm, or coxph containing a model whose coefficients are
                    the median of the coefficients of the bootstrapped models
boot.accuracy   The accuracy of the mboot.model model
boot.sensitivity The sensitivity of the mboot.model model
boot.specificity The specificity of the mboot.model model
z.NRIs          A matrix with the z-score of the NRI for each model term, estimated using the
                    bootstrap train sets
z.IDIs          A matrix with the z-score of the IDI for each model term, estimated using the
                    bootstrap train sets
test.z.NRIs     A matrix with the z-score of the NRI for each model term, estimated using the
                    bootstrap test sets
test.z.IDIs     A matrix with the z-score of the IDI for each model term, estimated using the
                    bootstrap test sets
NRI             A matrix with the NRI for each model term, estimated using the bootstrap test
                    sets
IDI             A matrix with the IDI for each model term, estimated using the bootstrap test
                    sets
testOutcome     A vector that contains all the individual outcomes used to validate the model in
                    the bootstrap test sets
testPrediction  A vector that contains all the individual predictions used to validate the model
                    in the bootstrap test sets

Author(s)

Jose G. Tamez-Pena and Antonio Martinez-Torteya

See Also

bootstrapValidation_Res, plot.bootstrapValidation_Bin, summary.bootstrapValidation_Bin
**bootstrapValidation_Res**

*Bootstrap validation of regression models*

**Description**

This function bootstraps the model \( n \) times to estimate for each variable the empirical bootstrapped distribution of model coefficients, and net residual improvement (NeRI). At each bootstrap the non-observed data is predicted by the trained model, and statistics of the test prediction are stored and reported.

**Usage**

```r
bootstrapValidation_Res(fraction = 1,
  loops = 200,
  model.formula, 
  Outcome, 
  data, 
  type = c("LM", "LOGIT", "COX"),
  plots = FALSE,
  bestmodel.formula=NULL)
```

**Arguments**

- `fraction` The fraction of data (sampled with replacement) to be used as train
- `loops` The number of bootstrap loops
- `model.formula` An object of class `formula` with the formula to be used
- `Outcome` The name of the column in `data` that stores the variable to be predicted by the model
- `data` A data frame where all variables are stored in different columns
- `type` Fit type: Logistic ("LOGIT"), linear ("LM"), or Cox proportional hazards ("COX")
- `plots` Logical. If `TRUE`, density distribution plots are displayed
- `bestmodel.formula` An object of class `formula` with the best formula to be compared

**Details**

The bootstrap validation will estimate the confidence interval of the model coefficients and the NeRI. It will also compute the train and blind test root-mean-square error (RMSE), as well as the distribution of the NeRI \( p \)-values.
Value

- **data**: The data frame used to bootstrap and validate the model.
- **outcome**: A vector with the predictions made by the model.
- **boot.model**: An object of class `lm`, `glm`, or `coxph` containing a model whose coefficients are the median of the coefficients of the bootstrapped models.
- **NeRIs**: A matrix with the NeRI for each model term, estimated using the bootstrap test sets.
- **tStudent.pvalues**: A matrix with the t-test p-value of the NeRI for each model term, estimated using the bootstrap train sets.
- **wilcox.pvalues**: A matrix with the Wilcoxon rank-sum test p-value of the NeRI for each model term, estimated using the bootstrap train sets.
- **bin.pvalues**: A matrix with the binomial test p-value of the NeRI for each model term, estimated using the bootstrap train sets.
- **F.pvalues**: A matrix with the F-test p-value of the NeRI for each model term, estimated using the bootstrap train sets.
- **test.tStudent.pvalues**: A matrix with the t-test p-value of the NeRI for each model term, estimated using the bootstrap test sets.
- **test.wilcox.pvalues**: A matrix with the Wilcoxon rank-sum test p-value of the NeRI for each model term, estimated using the bootstrap test sets.
- **test.bin.pvalues**: A matrix with the binomial test p-value of the NeRI for each model term, estimated using the bootstrap test sets.
- **test.F.pvalues**: A matrix with the F-test p-value of the NeRI for each model term, estimated using the bootstrap test sets.
- **testPrediction**: A vector that contains all the individual predictions used to validate the model in the bootstrap test sets.
- **testOutcome**: A vector that contains all the individual outcomes used to validate the model in the bootstrap test sets.
- **testResiduals**: A vector that contains all the residuals used to validate the model in the bootstrap test sets.
- **trainPrediction**: A vector that contains all the individual predictions used to validate the model in the bootstrap train sets.
- **trainOutcome**: A vector that contains all the individual outcomes used to validate the model in the bootstrap train sets.
- **trainResiduals**: A vector that contains all the residuals used to validate the model in the bootstrap train sets.
- **testRMSE**: The global RMSE, estimated using the bootstrap test sets.
- **trainRMSE**: The global RMSE, estimated using the bootstrap train sets.
**bootstrapVarElimination_Bin**

trainSampleRMSE  
A vector with the RMSEs in the bootstrap train sets

testSampledRMSE  
A vector with the RMSEs in the bootstrap test sets

**Author(s)**

Jose G. Tamez-Pena and Antonio Martinez-Torreya

**See Also**

bootstrapValidation_Bin, plot.bootstrapValidation_Res

---

**bootstrapVarElimination_Bin**  
IDI/NRI-based backwards variable elimination with bootstrapping

**Description**

This function removes model terms that do not improve the bootstrapped integrated discrimination improvement (IDI) or net reclassification improvement (NRI) significantly.

**Usage**

```r
bootstrapVarElimination_Bin(object,  
  pvalue = 0.05,  
  Outcome = "Class",  
  data,  
  startOffset = 0,  
  type = c("LOGIT", "LM", "COX"),  
  selectionType = c("zIDI", "zNRI"),  
  loops = 64,  
  print=TRUE,  
  plots=TRUE  
)
```

**Arguments**

- **object**  
  An object of class `lm`, `glm`, or `coxph` containing the model to be analyzed
- **pvalue**  
  The maximum *p*-value, associated to either IDI or NRI, allowed for a term in the model
- **Outcome**  
  The name of the column in `data` that stores the variable to be predicted by the model
- **data**  
  A data frame where all variables are stored in different columns
- **startOffset**  
  Only terms whose position in the model is larger than the `startOffset` are candidates to be removed
Fit type: Logistic ("LOGIT"), linear ("LM"), or Cox proportional hazards ("COX")

The type of index to be evaluated by the `improveProb` function (`Hmisc` package): $z$-score of IDI or of NRI

The number of bootstrap loops

Logical. If TRUE, information will be displayed

Logical. If TRUE, plots are displayed

For each model term $x_i$, the IDI or NRI is computed for the Full model and the reduced model (where the term $x_i$ removed). The term whose removal results in the smallest drop in bootstrapped improvement is selected. The hypothesis: the term adds classification improvement is tested by checking the p value of average improvement. If $p(IDIorNRI) > pvalue$, then the term is removed. In other words, only model terms that significantly aid in subject classification are kept. The procedure is repeated until no term fulfils the removal criterion.

An object of the same class as `object` containing the reduced model

The number of loops it took for the model to stabilize

A list with the NRI and IDI statistics of the reduced model, as given by the `getVarBin` function

An object of class `bootstrapValidation_Bin` containing the results of the bootstrap validation in the reduced model

An object of class `formula` with the formula used to fit the reduced model

The name of the last term that was removed (-1 if all terms were removed)

The model will have the fitted model that had close to maximum bootstrapped test accuracy

The formula of the model before False Selection Correction

the string formula of the model that had the best or close to the best test accuracy

Jose G. Tamez-Pena and Antonio Martinez-Torteya


See Also

`bootstrapVarElimination_Res`, `backVarElimination_Bin`, `backVarElimination_Res`
bootstrapVarElimination_Res

**NeRI-based backwards variable elimination with bootstrapping**

**Description**
This function removes model terms that do not improve the bootstrapped net residual improvement (NeRI) significantly.

**Usage**
```r
bootstrapVarElimination_Res(object, pvalue = 0.05, Outcome = "Class", data, startOffset = 0, type = c("LOGIT", "LM", "COX"), testType = c("Binomial", "Wilcox", "tStudent", "Ftest"), loops = 64, setIntersect = 1, print=TRUE, plots=TRUE )
```

**Arguments**
- `object`: An object of class `lm`, `glm`, or `coxph` containing the model to be analysed
- `pvalue`: The maximum p-value, associated to the NeRI, allowed for a term in the model
- `Outcome`: The name of the column in `data` that stores the variable to be predicted by the model
- `data`: A data frame where all variables are stored in different columns
- `startOffset`: Only terms whose position in the model is larger than the `startOffset` are candidates to be removed
- `type`: Fit type: Logistic ("LOGIT"), linear ("LM"), or Cox proportional hazards ("COX")
- `testType`: Type of non-parametric test to be evaluated by the `improvedResiduals` function: Binomial test ("Binomial"), Wilcoxon rank-sum test ("Wilcox"), Student’s t-test ("tStudent"), or F-test ("Ftest")
- `loops`: The number of bootstrap loops
- `setIntersect`: The intersect of the model (To force a zero intersect, set this value to 0)
- `print`: Logical. If TRUE, information will be displayed
- `plots`: Logical. If TRUE, plots are displayed
Details

For each model term $x_i$, the residuals are computed for the Full model and the reduced model (where the term $x_i$ removed). The term whose removal results in the smallest drop in bootstrapped test residuals improvement is selected. The hypothesis: the term improves residuals is tested by checking the p-value of average improvement. If $p(residuals_{better} than reduced residuals) > pvalue$, then the term is removed. In other words, only model terms that significantly aid in improving residuals are kept. The procedure is repeated until no term fulfills the removal criterion. The p-values of improvement can be computed via a sign-test (Binomial) a paired Wilcoxon test, paired t-test or f-test. The first three tests compare the absolute values of the residuals, while the f-test test if the variance of the residuals is improved significantly.

Value

- `back.model`: An object of the same class as `object` containing the reduced model
- `loops`: The number of loops it took for the model to stabilize
- `reclas.info`: A list with the NeRI statistics of the reduced model, as given by the `getVar.Res` function
- `bootCV`: An object of class `bootstrapValidation.Res` containing the results of the bootstrap validation in the reduced model
- `back.formula`: An object of class `formula` with the formula used to fit the reduced model
- `lastRemoved`: The name of the last term that was removed (-1 if all terms were removed)
- `at.opt.model`: The model with close to minimum bootstrapped RMSE
- `beforeFSC.formula`: The formula of the model before the FSC stage
- `at.RMSE.formula`: The string formula of the model that had the minimum or close to minimum RMSE

Author(s)

Jose G. Tamez-Pena and Antonio Martinez-Torteya

See Also

`bootstrapVarElimination_Bin`, `backVarElimination_Res`, `bootstrapValidation_Res`

Description

This function returns a set of models that best predict the outcome. Based on a Bootstrap Stage Wise Model Selection algorithm.
BSWiMS.model

Usage

BSWiMS.model(formula, data, 
    type = c("Auto","LM","LOGIT","COX"),
    testType = c("Auto","zIDI",
                 "zNRI",
                 "Binomial",
                 "Wilcox",
                 "tStudent",
                 "Ftest"),
    pvalue=0.05,
    variableList=NULL,
    size=0,
    loops=32,
    elimination.bootstrap.steps = 100,
    fraction=1.0,
    maxTrainModelSize=20,
    maxCycles=20,
    print=FALSE,
    plots=FALSE,
    featureSize=0,
    NumberOfRepeats=1
)

Arguments

 formula  An object of class formula with the formula to be fitted
data     A data frame where all variables are stored in different columns
type     The fit type. Auto will determine the fitting based on the formula
testType For an Binary-based optimization, the type of index to be evaluated by the 
improveProb function (Hmisc package): z-value of Binary or of NRI. For a 
NeRI-based optimization, the type of non-parametric test to be evaluated by the 
improvedResiduals function: Binomial test ("Binomial"), Wilcoxon rank-sum 
test ("Wilcox"), Student’s t-test ("tStudent"), or F-test ("Ftest")
pvalue   The maximum p-value, associated to the testType, allowed for a term in the 
          model (it will control the false selection rate)
variableList A data frame with two columns. The first one must have the names of the can-
              didate variables and the other one the description of such variables
size      The number of candidate variables to be tested (the first size variables from 
          variableList)
loops     The number of bootstrap loops for the forward selection procedure
elimination.bootstrap.steps The number of bootstrap loops for the backwards elimination procedure
fraction  The fraction of data (sampled with replacement) to be used as train
maxTrainModelSize
   Maximum number of terms that can be included in the each forward selection model
maxCycles
   The maximum number of model generation cycles
print
   Logical. If TRUE, information will be displayed
plots
   Logical. If TRUE, plots are displayed
featureSize
   The original number of features to be explored in the data frame.
NumberOfRepeats
   How many times the BSWiMS search will be repeated

Details
This is a core function of FRESA.CAD. The function will generate a set of B:SWiMS models from the data based on the provided baseline formula. The function will loop extracting a models whose all terms are statistical significant. After each loop it will remove the significant terms, and it will repeat the model generation until no mode significant models are found or the maximum number of cycles is reached.

Value

   BSWiMS.model     the output of the bootstrap backwards elimination step
   forward.model    The output of the forward selection step
   update.model     The output of the forward selection step
   univariate       The univariate ranking of variables if no list of features was provided
   bagging          The model after bagging the set of models
   formula.list     The formulas extracted at each cycle
   forward.selection.list
                   All formulas generated by the forward selection procedure
   oridinalModels   A list of scores, the data and a formulas vector required for ordinal scores predictions

Author(s)
Jose G. Tamez-Pena

References
Examples

```r
## Not run:

# Start the graphics device driver to save all plots in a pdf format
data(file = "BSWiMS.model.Example.pdf", width = 8, height = 6)

# Get the stage C prostate cancer data from the rpart package
data(stagec, package = "rpart")
options(NA.action = "na.pass")
stagec_mat <- cbind(pgstat = stagec$pgstat,
pctime = stagec$pctime,
as.data.frame(model.matrix(Surv(pctime, pgstat) ~ ., stagec))[-1])
fnames <- colnames(stagec_mat)
fnames <- str_replace_all(fnames, ":","\_\_")
colnames(stagec_mat) <- fnames
dataCancerImputed <- nearestNeighborImpute(stagec_mat)

# Get a Cox proportional hazards model using:
# - The default parameters
md <- BSWiMS.model(formula = Surv(pctime, pgstat) ~ 1,
data = dataCancerImputed)

# Plot the bootstrap validation
pr <- plot(md$BSWiMS.model$bootcv)

# Get the coefficients summary
sm <- summary(md)
print(sm$coefficients)

# Plot the bagged model
pl <- plotModels.ROC(cbind(dataCancerImputed$pgstat,
predict(md, dataCancerImputed)),
main = "Bagging Predictions")

# Get a Cox proportional hazards model using:
# - The default parameters but repeated 10 times
md <- BSWiMS.model(formula = Surv(pctime, pgstat) ~ 1,
data = dataCancerImputed,
NumberOfRepeats = 10)

# Get the coefficients summary
sm <- summary(md)
print(sm$coefficients)

# Check all the formulas
print(md$formula.list)

# Plot the bagged model
pl <- plotModels.ROC(cbind(dataCancerImputed$pgstat,
predict(md, dataCancerImputed)),
main = "Bagging Predictions")
```
main = "Bagging Predictions"

# Get a regression of the survival time

t imeSubjects <- dataCancerImputed
t imeSubjects$pgtime <- log(timeSubjects$pgtime)

md <- BSWiMS.model(formula = pgtime ~ 1,
data = timeSubjects,
                      )
pt <- plot(md$BSWiMS.model$bootCV)
sm <- summary(md)
print(sm$coefficients)

# Get a logistic regression model using the default parameters and removing time as possible predictor

data(stagec, package = "rpart")
stagec$pgtime <- NULL
stagec_mat <- cbind(pgstat = stagec$pgstat,
as.data.frame(model.matrix(pgstat ~ .*., stagec))[-1])
fnames <- colnames(stagec_mat)
fnames <- str_replace_all(fnames, ":", "_")
colnames(stagec_mat) <- fnames
dataCancerImputed <- nearestNeighborImpute(stagec_mat)

md <- BSWiMS.model(formula = pgstat ~ 1,
data = dataCancerImputed)
pt <- plot(md$BSWiMS.model$bootCV)
sm <- summary(md)
print(sm$coefficients)

# Get an ordinal regression of grade model using GBSG2 data
# - The default parameters and removing the time and status as possible predictor

data("GBSG2", package = "TH.data")

# Prepare the model frame for prediction
GBSG2$time <- NULL;
GBSG2$cens <- NULL;
GBSG2_mat <- cbind(tgrade = as.numeric(GBSG2$tgrade),
as.data.frame(model.matrix(tgrade~.*., GBSG2))[-1])

fnames <- colnames(GBSG2_mat)
fnames <- str_replace_all(fnames, ":", "_")
colnames(GBSG2_mat) <- fnames

md <- BSWiMS.model(formula = tgrade ~ 1,
data = GBSG2_mat)
cancervarnames

Data frame used in several examples of this package

Description

This data frame contains two columns, one with names of variables, and the other with descriptions of such variables. It is used in several examples of this package. Specifically, it is used in examples working with the stage C prostate cancer data from the rpart package.

Usage

data(cancervarnames)

Format

A data frame with names and descriptions of the variables used in several examples

Var A column with the names of the variables
Description A column with a short description of the variables

Examples

data(cancervarnames)
crossValidationFeatureSelection_Bin

IDI/NRI-based selection of a linear, logistic, or Cox proportional hazards regression model from a set of candidate variables

Description

This function performs a cross-validation analysis of a feature selection algorithm based on the integrated discrimination improvement (IDI) or the net reclassification improvement (NRI) to return a predictive model. It is composed of an IDI/NRI-based feature selection followed by an update procedure, ending with a bootstrapping backwards feature elimination. The user can control how many train and blind test sets will be evaluated.

Usage

crossValidationFeatureSelection_Bin(size = 10, 
fraction = 1.0, 
pvalue = 0.05, 
loops = 100, 
covariates = "1", 
Outcome, 
timeOutcome = "Time", 
variableList, 
data, 
maxTrainModelSize = 20, 
type = c("LM", "LOGIT", "COX"), 
selectionType = c("zIDI", "zNRI"), 
startOffset = 0, 
elimination.bootstrap.steps = 100, 
trainFraction = 0.67, 
trainRepetition = 9, 
bootstrap.steps = 100, 
nk = 0, 
unirank = NULL, 
print=TRUE, 
plots=TRUE, 
lambda="lambda.1se", 
equivalent=FALSE, 
bswimsCycles=10, 
usrFitFun=NULL, 
featureSize=0)

Arguments

size The number of candidate variables to be tested (the first size variables from variableList)

fraction The fraction of data (sampled with replacement) to be used as train
pvalue The maximum $p$-value, associated to either IDI or NRI, allowed for a term in the model
loops The number of bootstrap loops
covariates A string of the type "1 + var1 + var2" that defines which variables will always be included in the models (as covariates)
Outcome The name of the column in data that stores the variable to be predicted by the model
timeOutcome The name of the column in data that stores the time to event (needed only for a Cox proportional hazards regression model fitting)
variableList A data frame with two columns. The first one must have the names of the candidate variables and the other one the description of such variables
data A data frame where all variables are stored in different columns
maxTrainModelSize Maximum number of terms that can be included in the model
type Fit type: Logistic ("LOGIT"), linear ("LM"), or Cox proportional hazards ("COX")
selectionType The type of index to be evaluated by the improveProb function (Hmisc package): z-score of IDI or of NRI
startOffset Only terms whose position in the model is larger than the startOffset are candidates to be removed
elimination.bootstrap.steps The number of bootstrap loops for the backwards elimination procedure
trainFraction The fraction of data (sampled with replacement) to be used as train for the cross-validation procedure
trainRepetition The number of cross-validation folds (it should be at least equal to $1/trainFraction$ for a complete cross-validation)
bootstrap.steps The number of bootstrap loops for the confidence intervals estimation
nk The number of neighbours used to generate a $k$-nearest neighbours (KNN) classification. If zero, $k$ is set to the square root of the number of cases. If less than zero, it will not perform the KNN classification
unirank A list with the results yielded by the uniRankVar function, required only if the rank needs to be updated during the cross-validation procedure
print Logical. If TRUE, information will be displayed
plots Logical. If TRUE, plots are displayed
lambda The passed value to the s parameter of the glmnet cross validation coefficient
equivalent Is set to TRUE CV will compute the equivalent model
bswimsCycles The maximum number of models to be returned by BSwiMS.model
usrFitFun A user fitting function to be evaluated by the cross validation procedure
featureSize The original number of features to be explored in the data frame.
crossValidationFeatureSelection_Bin

Details
This function produces a set of data and plots that can be used to inspect the degree of over-fitting or shrinkage of a model. It uses bootstrapped data, cross-validation data, and, if possible, retrain data. During each cycle, a train and a test ROC will be generated using bootstrapped data. At the end of the cross-validation feature selection procedure, a set of three plots may be produced depending on the specifications of the analysis. The first plot shows the ROC for each cross-validation blind test. The second plot, if enough samples are given, shows the ROC of each model trained and tested in the blind test partition. The final plot shows ROC curves generated with the train, the bootstrapped blind test, and the cross-validation test data. Additionally, this plot will also contain the ROC of the cross-validation mean test data, and of the cross-validation coherence. These set of plots may be used to get an overall perspective of the expected model shrinkage. Along with the plots, the function provides the overall performance of the system (accuracy, sensitivity, and specificity). The function also produces a report of the expected performance of a KNN algorithm trained with the selected features of the model, and an elastic net algorithm. The test predictions obtained with these algorithms can then be compared to the predictions generated by the logistic, linear, or Cox proportional hazards regression model.

Value

formula.list  A list containing objects of class formula with the formulas used to fit the models found at each cycle

Models.testPrediction  A data frame with the blind test set predictions (Full B:SWiMS,Median,Bagged,Forward,Backwards Eliminations) made at each fold of the cross validation, where the models used to generate such predictions (formula.list) were generated via a feature selection process which included only the train set. It also includes a column with the outcome of each prediction, and a column with the number of the fold at which the prediction was made.

FullBSWiMS.testPrediction  A data frame similar to Models.testPrediction, but where the model used to generate the predictions was the Full model, generated via a feature selection process which included all data.

TestRetrained.blindPredictions  A data frame similar to Models.testPrediction, but where the models were retrained on an independent set of data (only if enough samples are given at each fold)

LastTrainBSWiMS.bootstrapped  An object of class bootstrapValidation_Bin containing the results of the bootstrap validation in the last trained model

Test.accuracy  The global blind test accuracy of the cross-validation procedure

Test.sensitivity  The global blind test sensitivity of the cross-validation procedure

Test.specificity  The global blind test specificity of the cross-validation procedure

Train.correlationsToFull  The Spearman $\rho$ rank correlation coefficient between the predictions made with each model from formula.list and the Full model in the train set
Blind.correlationsToFull
The Spearman $\rho$ rank correlation coefficient between the predictions made with each model from formula.list and the Full model in the test set

FullModelAtFoldAccuracies
The blind test accuracy for the Full model at each cross-validation fold

FullModelAtFoldSpecificities
The blind test specificity for the Full model at each cross-validation fold

FullModelAtFoldSensitivities
The blind test sensitivity for the Full model at each cross-validation fold

FullModelAtFoldAUC
The blind test ROC AUC for the Full model at each cross-validation fold

AtCVFoldModelBlindAccuracies
The blind test accuracy for the Full model at each final cross-validation fold

AtCVFoldModelBlindSpecificities
The blind test specificity for the Full model at each final cross-validation fold

AtCVFoldModelBlindSensitivities
The blind test sensitivity for the Full model at each final cross-validation fold

CVTrain.Accuracies
The train accuracies at each fold

CVTrain.Sensitivity
The train sensitivity at each fold

CVTrain.Question
The train specificity at each fold

CVTrain.AUCs
The train ROC AUC for each fold

forwardSelection
A list containing the values returned by ForwardSelection.Model.Bin using all data

updateforwardSelection
A list containing the values returned by updateModel.Bin using all data and the model from ForwardSelection

BSWiMS
A list containing the values returned by bootstrapVarElimination_Bin using all data and the model from updateforwardSelection

FullBSWiMS.bootstrapped
An object of class bootstrapValidation_Bin containing the results of the bootstrap validation in the Full model

Models.testSensitivities
A matrix with the mean ROC sensitivities at certain specificities for each train and all test cross-validation folds using the cross-validation models (i.e. 0.95, 0.90, 0.80, 0.70, 0.60, 0.50, 0.40, 0.30, 0.20, 0.10, and 0.05)

FullKNN.testPrediction
A data frame similar to Models.testPrediction, but where a KNN classifier with the same features as the Full model was used to generate the predictions

KNN.testPrediction
A data frame similar to Models.testPrediction, but where KNN classifiers with the same features as the cross-validation models were used to generate the predictions at each cross-validation fold
An object of class `cv.glmnet` containing the results of an elastic net cross-validation fit

A data frame similar to `Models.testPrediction`, but where the predictions were made by the elastic net model

A list with the elastic net Full model and the models found at each cross-validation fold

The list of accuracies of an univariate analysis on each one of the model variables in the train sets

The list of accuracies of an univariate analysis on each one of the model variables in the test sets

The accuracy coherence of the top ranked variable on the test set

The accuracy coherence of the top ranked variable on the train set

A data frame with the outcome and the train prediction of every model

A data frame with the outcome and the train prediction at each CV fold for the main model

A data frame with the outcome and the prediction of each enet lasso model

The ensemble prediction by all models on the test data

The list of formulas with "optimal" performance

The list of formulas produced by the forward procedure

The list of the bagged models

The list of variables used by LASSO fitting

Jose G. Tamez-Pena and Antonio Martinez-Torteya


See Also

crossValidationFeatureSelection_Res

**Description**

This function performs a cross-validation analysis of a feature selection algorithm based on net residual improvement (NeRI) to return a predictive model. It is composed of a NeRI-based feature selection followed by an update procedure, ending with a bootstrapping backwards feature elimination. The user can control how many train and blind test sets will be evaluated.

**Usage**

```r
crossValidationFeatureSelection_Res(size = 10,
    fraction = 1.0,
    pvalue = 0.05,
    loops = 100,
    covariates = "1",
    Outcome,
    timeOutcome = "Time",
    variableList,
    data,
    maxTrainModelSize = 20,
    type = c("LM", "LOGIT", "COX"),
    testType = c("Binomial",
    "Wilcox",
    "tStudent",
    "Ftest"),
    startOffset = 0,
    elimination.bootstrap.steps = 100,
    trainFraction = 0.67,
    trainRepetition = 9,
    setIntersect = 1,
    unirank = NULL,
    print=TRUE,
    plots=TRUE,
    lambda="lambda.1se",
    equivalent=FALSE,
    bswimsCycles=10,
    usrFitFun=NULL,
    featureSize=0)
```

**Arguments**

- `size` The number of candidate variables to be tested (the first `size` variables from `variableList`)
fraction | The fraction of data (sampled with replacement) to be used as train
pvalue | The maximum $p$-value, associated to the NeRI, allowed for a term in the model
loops | The number of bootstrap loops
covariates | A string of the type "$1 + var1 + var2" that defines which variables will always be included in the models (as covariates)
Outcome | The name of the column in data that stores the variable to be predicted by the model
timeOutcome | The name of the column in data that stores the time to event (needed only for a Cox proportional hazards regression model fitting)
variableList | A data frame with two columns. The first one must have the names of the candidate variables and the other one the description of such variables
data | A data frame where all variables are stored in different columns
maxTrainModelSize | Maximum number of terms that can be included in the model
type | Fit type: Logistic ("LOGIT"), linear ("LM"), or Cox proportional hazards ("COX")
testType | Type of non-parametric test to be evaluated by the improvedResiduals function: Binomial test ("Binomial"), Wilcoxon rank-sum test ("Wilcox"), Student’s $t$-test ("tStudent"), or $F$-test ("Ftest")
startOffset | Only terms whose position in the model is larger than the startOffset are candidates to be removed
elimination.bootstrap.steps | The number of bootstrap loops for the backwards elimination procedure
trainFraction | The fraction of data (sampled with replacement) to be used as train for the cross-validation procedure
setIntersect | The intersect of the model (To force a zero intersect, set this value to 0)
trainRepetition | The number of cross-validation folds (it should be at least equal to $1/trainFraction$ for a complete cross-validation)
unirank | A list with the results yielded by the uniRankVar function, required only if the rank needs to be updated during the cross-validation procedure
print | Logical. If TRUE, information will be displayed
plots | Logical. If TRUE, plots are displayed
lambda | The passed value to the s parameter of the glmnet cross validation coefficient
equivalent | Is set to TRUE CV will compute the equivalent model
bswimsCycles | The maximum number of models to be returned by BSWiMS.model
usrfitfun | A user fitting function to be evaluated by the cross validation procedure
featuresize | The original number of features to be explored in the data frame.

**Details**

This function produces a set of data and plots that can be used to inspect the degree of over-fitting or shrinkage of a model. It uses bootstrapped data, cross-validation data, and, if possible, retrain data.
Value

formulanlist  A list containing objects of class `formula` with the formulas used to fit the models found at each cycle

Models.testPrediction  A data frame with the blind test set predictions made at each fold of the cross-validation (Full B: SWiMS, Median, Bagged, Forward, Backward Elimination), where the models used to generate such predictions (formulanlist) were generated via a feature selection process which included only the train set. It also includes a column with the outcome of each prediction, and a column with the number of the fold at which the prediction was made.

FullBSWiMS.testPrediction  A data frame similar to Models.testPrediction, but where the model used to generate the predictions was the Full model, generated via a feature selection process which included all data.

BSWiMS  A list containing the values returned by bootstrapVarElimination.Res using all data and the model from updatedforwardModel

forwardSelection  A list containing the values returned by ForwardSelection.Model.Res using all data

updatedforwardModel  A list containing the values returned by updateModel.Res using all data and the model from forwardSelection

testrMSE  The global blind test root-mean-square error (RMSE) of the cross-validation procedure

testPearson  The global blind test Pearson $r$ product-moment correlation coefficient of the cross-validation procedure

testSpearman  The global blind test Spearman $\rho$ rank correlation coefficient of the cross-validation procedure

FulltestRMSE  The global blind test RMSE of the Full model

FullTestPearson  The global blind test Pearson $r$ product-moment correlation coefficient of the Full model

FullTestSpearman  The global blind test Spearman $\rho$ rank correlation coefficient of the Full model

trainRMSE  The train RMSE at each fold of the cross-validation procedure

trainPearson  The train Pearson $r$ product-moment correlation coefficient at each fold of the cross-validation procedure

trainSpearman  The train Spearman $\rho$ rank correlation coefficient at each fold of the cross-validation procedure

FullTrainRMSE  The train RMSE of the Full model at each fold of the cross-validation procedure

FullTrainPearson  The train Pearson $r$ product-moment correlation coefficient of the Full model at each fold of the cross-validation procedure
**crossValidationFeatureSelection_Res**

- **FullTrainSpearman**: The train Spearman $\rho$ rank correlation coefficient of the Full model at each fold of the cross-validation procedure.

- **testRMSEAtFold**: The blind test RMSE at each fold of the cross-validation procedure.

- **FullTestRMSEAtFold**: The blind test RMSE of the Full model at each fold of the cross-validation procedure.

- **Fullenent**: An object of class `cv.glmnet` containing the results of an elastic net cross-validation fit.

- **LASSO.testPredictions**: A data frame similar to `Models.testPrediction`, but where the predictions were made by the elastic net model.

- **LASSOVariables**: A list with the elastic net Full model and the models found at each cross-validation fold.

- **byFoldTestMS**: A vector with the Mean Square error for each blind fold.

- **byFoldTestSpearman**: A vector with the Spearman correlation between prediction and outcome for each blind fold.

- **byFoldTestPearson**: A vector with the Pearson correlation between prediction and outcome for each blind fold.

- **byFoldCstat**: A vector with the C-index (Somers’ Dxy rank correlation : `rcorr.cens`) between prediction and outcome for each blind fold.

- **CVBlindPearson**: A vector with the Pearson correlation between the outcome and prediction for each repeated experiment.

- **CVBlindSpearman**: A vector with the Spearman correlation between the outcome and prediction for each repeated experiment.

- **CVBlindRMS**: A vector with the RMS between the outcome and prediction for each repeated experiment.

- **Models.trainPrediction**: A data frame with the outcome and the train prediction of every model.

- **FullBSWiMS.trainPrediction**: A data frame with the outcome and the train prediction at each CV fold for the main model.

- **LASSO.trainPredictions**: A data frame with the outcome and the prediction of each enet lasso model.

- **uniTrainMSS**: A data frame with mean square of the train residuals from the univariate models of the model terms.

- **uniTestMSS**: A data frame with mean square of the test residuals of the univariate models of the model terms.

- **BSWiMS.ensemble.prediction**: The ensemble prediction by all models on the test data.

- **AtOptFormulas.list**: The list of formulas with "optimal" performance.
CVsignature

ForwardFormulas.list
   The list of formulas produced by the forward procedure
baggFormulas.list
   The list of the bagged models
LassoFilterVarList
   The list of variables used by LASSO fitting

Author(s)
Jose G. Tamez-Pena and Antonio Martinez-Torteya

See Also
   crossValidationFeatureSelection_Bin, improvedResiduals, bootstrapVarElimination_Res

---

CVsignature  Cross-validated Signature

Description
A formula based wrapper of the getSignature function

Usage
CVsignature(formula = formula, data=NULL,...)

Arguments
   formula     The base formula
   data        The data to be used for training the signature method
   ...         Parameters for the getSignature function

Value
   fit         A getSignature object.
   method      The distance method
   variable.importance
                The named vector of relevant features

Author(s)
Jose G. Tamez-Pena

See Also
   getSignature, signatureDistance
EmpiricalSurvDiff

Estimate the LR value and its associated p-values

Description

Permutations or Bootstrapping computation of the standardized log-rank (SLR) or the Chi=SLR^2 p-values for differences in survival times

Usage

EmpiricalSurvDiff(times=times,
status=status,
groups=groups,
samples=1000,
type=c("SLR","Chi"),
plots=FALSE,
minAproxSamples=100,
computeDist=FALSE,
...
)

Arguments

- **times**: A numeric vector with the observed times to event
- **status**: A numeric vector indicating if the time to event is censored
- **groups**: A numeric vector indicating the label of the two survival groups
- **samples**: The number of bootstrap samples
- **type**: The type of log-rank statistics. SLR or Chi
- **plots**: If TRUE, the Kaplan-Meier plot will be plotted
- **minAproxSamples**: The number of tail samples used for the normal-distribution approximation
- **computeDist**: If TRUE, it will compute the bootstrapped distribution of the SLR
- **...**: Additional parameters for the plot

Details

It will compute the null distribution of the SRL or the square SLR (Chi) via permutations, and it will return the p-value of differences between survival times between two groups. It may also be used to compute the empirical distribution of the difference in SLR using bootstrapping. (computeDist=TRUE) The p-values will be estimated based on the sampled distribution, or normal-approximated along the tails.
EmpiricalSurvDiff

Value

- **pvalue**: the minimum one-tailed p-value: \( \min[p(SRL < 0), p(SRL > 0)] \) for type="SLR"
or the two tailed p-value: \( 1 - p(|SRL| > 0) \) for type="Chi"

- **LR**: A list of LR statistics: LR=Expected, VR=Variance, SLR=Standardized LR.

- **p.equal**: The two tailed p-value: \( 1 - p(|SRL| > 0) \)

- **p.sup**: The one tailed p-value: \( p(SRL < 0) \), return NA for type="Chi"

- **p.inf**: The one tailed p-value: \( p(SRL > 0) \), return NA for type="Chi"

- **nullDist**: permutation derived probability density function of the null distribution

- **LRDist**: bootstrapped derived probability density function of the SLR (computeDist=TRUE)

Author(s)

Jose G. Tamez-Pena

Examples

```
## Not run:

library(rpart)
data(stagec)

# The Log-Rank Analysis using survdiff

lrsurvdiff <- survdiff(Surv(pgti, pstat)-grade>2, data=stagec)
print(lrsurvdiff)

# The Log-Rank Analysis: permutations of the null Chi distribution

lrp <- EmpiricalSurvDiff(stagec$ptgtime, stagec$pstat, stagec$grade>2,
                          type="Chi", plots=TRUE, samples=10000,
                          main="Chi Null Distribution")
print(list(unlist(c(lrp$LR, lrp$pvalue)))))

# The Log-Rank Analysis: permutations of the null SLR distribution

lrp <- EmpiricalSurvDiff(stagec$ptgtime, stagec$pstat, stagec$grade>2,
                          type="SLR", plots=TRUE, samples=10000,
                          main="SLR Null Distribution")
print(list(unlist(c(lrp$LR, lrp$pvalue)))))

# The Log-Rank Analysis: Bootstraping the SLR distribution

lrp <- EmpiricalSurvDiff(stagec$ptgtime, stagec$pstat, stagec$grade>2,
                          computeDist=TRUE, plots=TRUE, samples=100000,
                          main="SLR Null and SLR bootrapped")
print(list(unlist(c(lrp$LR, lrp$pvalue)))))

## End(Not run)
```
ensemblePredict

The median prediction from a list of models

Description

Given a list of model formulas, this function will train such models and return the single(ensemble) prediction from the list of formulas on a test data set. It may also provide a $k$-nearest neighbors (KNN) prediction using the features listed in such models.

Usage

```r
ensemblePredict(formulaList, 
    trainData, 
    testData = NULL, 
    predictType = c("prob", "linear"), 
    type = c("LOGIT", "LM", "COX","SVM"), 
    Outcome = NULL, 
    nk = 0)
```

Arguments

- `formulaList`: A list made of objects of class `formula`, each representing a model formula to be fitted and predicted with
- `trainData`: A data frame with the data to train the model, where all variables are stored in different columns
- `testData`: A data frame similar to `trainData`, but with the data set to be predicted. If `NULL`, `trainData` will be used
- `predictType`: Prediction type: Probability ("prob") or linear predictor ("linear")
- `type`: Fit type: Logistic ("LOGIT"), linear ("LM"), or Cox proportional hazards ("COX")
- `Outcome`: The name of the column in data that stores the variable to be predicted by the model
- `nk`: The number of neighbors used to generate the KNN classification. If zero, $k$ is set to the square root of the number of cases. If less than zero, it will not perform the KNN classification

Value

- `ensemblePredict`: A vector with the median prediction for the `testData` data set, using the models from `formulaList`
- `medianKNNPredict`: A vector with the median prediction for the `testData` data set, using the KNN models
- `predictions`: A matrix, where each column represents the predictions made with each model from `formulaList`
KNN predictions: A matrix, where each column represents the predictions made with a different KNN model.

wPredict: A vector with the weighted mean ensemble.

Author(s)

Jose G. Tamez-Pena and Antonio Martinez-Torteya

Description

This function fits the candidate variables to the provided model, for each strata, on a control population. If the variance of the residual (the fitted observation minus the real observation) is reduced significantly, then, such residual is used in the resulting data frame. Otherwise, the control mean is subtracted to the observation.

Usage

featureAdjustment(variableList, 
  baseModel, 
  strata = NA, 
  data, 
  referenceframe, 
  type = c("LM", "GLS"), 
  pvalue = 0.05, 
  correlationGroup = "ID")

Arguments

variableList: A data frame with two columns. The first one must have the names of the candidate variables and the other one the description of such variables.

baseModel: A string of the type "1 + var1 + var2" that defines the model to which variables will be fitted.

strata: The name of the column in data that stores the variable that will be used to stratify the model.

data: A data frame where all variables are stored in different columns.

referenceframe: A data frame similar to data, but with only the control population.

type: Fit type: linear fitting ("LM"), or generalized least squares fitting ("GLS").

pvalue: The maximum p-value, associated to the F-test, for the model to be allowed to reduce variability.

correlationGroup: The name of the column in data that stores the variable to be used to group the data (only needed if type defined as "GLS")
Value
A data frame, where each input observation has been adjusted from data at each strata

Note
This function prints the residuals and the \( F \)-statistic for all candidate variables

Author(s)
Jose G. Tamez-Pena and Antonio Martinez-Torteya

---

**FilterUnivariate**

**Univariate Filters**

**Description**
Returns the top set of features that are statistically associated with the outcome.

**Usage**

```r
univariate_Logit(data=NULL, Outcome=NULL, pvalue=0.2, adjustMethod="BH",
                 uniTest=c("zIDI","zNRI"), limit=0,...)
univariate_residual(data=NULL, Outcome=NULL, pvalue=0.2, adjustMethod="BH",
                    uniTest=c("Ftest","Binomial","Wilcoxon","tStudent"),
                    type=c("LM","LOGIT"), limit=0,...)
univariate_tstudent(data=NULL, Outcome=NULL, pvalue=0.2, adjustMethod="BH",
                    limit=0,...)
univariate_Wilcoxon(data=NULL, Outcome=NULL, pvalue=0.2, adjustMethod="BH",
                    limit=0,...)
univariate_correlation(data=NULL, Outcome=NULL, pvalue=0.2, adjustMethod="BH",
                       method = "kendall", limit=0,...)
correlated_Remove(data=NULL, fnames=NULL, thr=0.999)
```

**Arguments**

- **data** The data frame
- **Outcome** The outcome feature
- **pvalue** The threshold \( p \)value used after the \( p \).adjust method
- **adjustMethod** The method used by the \( p \).adjust method
- **uniTest** The \( \text{unitTest} \) to be performed by the linear fitting model
- **type** The type of linear model: LM or LOGIT
- **method** The correlation method: pearson, spearman or kendall.
- **limit** The samples-wise fraction of features to return.
FilterUnivariate

fnames  The list of features to test inside the correlated_Remove function
thr  The maximum correlation to allow between features
...  Parameters to be passed to the correlated_Remove function

Value

Named vector with the adjusted p-values or the list of no-correlated features for the correlated_Remove

Author(s)

Jose G. Tamez-Pena

Examples

```r
## Not run:

library("FRESA.CAD")

### Univariate Filter Examples ###

# Get the stage C prostate cancer data from the rpart package
data(stagec, package = "rpart")

# Prepare the data. Create a model matrix without the event time and interactions
stagec$pgtime <- NULL
stagec$et <- as.factor(stagec$et)
options(na.action = 'na.pass')
stagec_mat <- cbind(pgtest = stagec$pgstat,
as.data.frame(model.matrix(pgtest ~ ., stagec))[-1])
fnames <- colnames(stagec_mat)
fnames <- str_replace_all(fnames, ":", "__")

# Impute the missing data
dataCancerImputed <- nearestNeighborImpute(stagec_mat)
dataCancerImputed[,1:ncol(dataCancerImputed)] <- sapply(dataCancerImputed, as.numeric)

# Get the top Features associated to pgstat
q_values <- univariate_Logit(data = dataCancerImputed,
Outcome = "pgstat",
pvalue = 0.05)

qValueMatrix <- q_values
idiqValueMatrix <- q_values
barplot(-log(q_values), las=2, cex.names=0.4, ylab="-log(Q)",
main="Association with PGStat: IDI Test")

q_values <- univariate_Logit(data = dataCancerImputed,
Outcome = "pgstat",
unitest = "zNRI", pvalue = 0.05)
qValueMatrix <- cbind(idiqValueMatrix, q_values[names(idiqValueMatrix)])
```
```r
q_values <- univariate_residual(data=dataCancerImputed,
     Outcome="pgstat",
     pvalue = 0.05,type="LOGIT")
qValueMatrix <- cbind(qValueMatrix,q_values[names(idiQValueMatrix)])

q_values <- univariate_tstudent(data=dataCancerImputed,
     Outcome="pgstat",
     pvalue = 0.05)
qValueMatrix <- cbind(qValueMatrix,q_values[names(idiQValueMatrix)])

q_values <- univariate_Wilcoxon(data=dataCancerImputed,
     Outcome="pgstat",
     pvalue = 0.05)
qValueMatrix <- cbind(qValueMatrix,q_values[names(idiQValueMatrix)])

q_values <- univariate_correlation(data=dataCancerImputed,
     Outcome="pgstat",
     pvalue = 0.05
   method = "pearson")

qValueMatrix <- cbind(qValueMatrix,q_values[names(idiQValueMatrix)])

# The qValueMatrix has the qValues of all filter methods.
colnames(qValueMatrix) <- c("IDI","NRI","F","t","W","K","P")

# Do the log transform to display the heatmap
qValueMatrix <- -log10(qValueMatrix)

# the Heatmap of the q-values
gplots::heatmap.2(qValueMatrix,Rowv = FALSE,dendrogram = "col",
     main = "Method q.values",cexRow = 0.4)

## End(Not run)
```

**ForwardSelection.Model.Bin**

*IDINRI-based feature selection procedure for linear, logistic, and Cox proportional hazards regression models*

**Description**

This function performs a bootstrap sampling to rank the variables that statistically improve prediction. After the frequency rank, the function uses a forward selection procedure to create a final model, whose terms all have a significant contribution to the integrated discrimination improvement (IDI) or the net reclassification improvement (NRI). For each bootstrap, the IDI/NRI is computed and the variable with the largest statically significant IDI/NRI is added to the model. The procedure
is repeated at each bootstrap until no more variables can be inserted. The variables that enter the model are then counted, and the same procedure is repeated for the rest of the bootstrap loops. The frequency of variable-inclusion in the model is returned as well as a model that uses the frequency of inclusion.

Usage

```r
ForwardSelection.Model.Bin(size = 100,
  fraction = 1,
  pvalue = 0.05,
  loops = 100,
  covariates = "1",
  Outcome,
  variableList,
  data,
  maxTrainModelSize = 20,
  type = c("LM", "LOGIT", "COX"),
  timeOutcome = "Time",
  selectionType=c("zIDI", "zNRI"),
  cores = 4,
  randsize = 0,
  featureSize=0)
```

Arguments

- **size**: The number of candidate variables to be tested (the first `size` variables from `variableList`)
- **fraction**: The fraction of data (sampled with replacement) to be used as train
- **pvalue**: The maximum $p$-value, associated to either IDI or NRI, allowed for a term in the model
- **loops**: The number of bootstrap loops
- **covariates**: A string of the type "$1 + \text{var1} + \text{var2}$" that defines which variables will always be included in the models (as covariates)
- **Outcome**: The name of the column in `data` that stores the variable to be predicted by the model
- **variableList**: A data frame with two columns. The first one must have the names of the candidate variables and the other one the description of such variables
- **data**: A data frame where all variables are stored in different columns
- **maxTrainModelSize**: Maximum number of terms that can be included in the model
- **type**: Fit type: Logistic ("LOGIT"), linear ("LM"), or Cox proportional hazards ("COX")
- **timeOutcome**: The name of the column in `data` that stores the time to event (needed only for a Cox proportional hazards regression model fitting)
- **selectionType**: The type of index to be evaluated by the `improveProb` function (`Hmisc` package): $z$-score of IDI or of NRI
- **cores**: Cores to be used for parallel processing
ForwardSelection.Model.Res

randsize is the model size of a random outcome. If randsize is less than zero, it will estimate the size.

featuresize is the original number of features to be explored in the data frame.

Value

final.model is an object of class lm, glm, or coxph containing the final model.

var.names is a vector with the names of the features that were included in the final model.

formula is an object of class formula with the formula used to fit the final model.

ranked.var is an array with the ranked frequencies of the features.

z.selection is a vector in which each term represents the z-score of the index defined in selectionType obtained with the Full model and the model without one term.

formula.list is a list containing objects of class formula with the formulas used to fit the models found at each cycle.

variableList is a list of variables used in the forward selection.

Author(s)

Jose G. Tamez-Pena and Antonio Martinez-Torteya

References


See Also

ForwardSelection.Model.Res

Description

This function performs a bootstrap sampling to rank the most frequent variables that statistically aid the models by minimizing the residuals. After the frequency rank, the function uses a forward selection procedure to create a final model, whose terms all have a significant contribution to the net residual improvement (NeRI).
Usage

```r
ForwardSelection.Model.Res(size = 100,
  fraction = 1,
  pvalue = 0.05,
  loops = 100,
  covariates = "1",
  Outcome,
  variableList,
  data,
  maxTrainModelSize = 20,
  type = c("LM", "LOGIT", "COX"),
  testType = c("Binomial", "Wilcoxon", "tStudent", "Ftest"),
  timeOutcome = "Time",
  cores = 4,
  randsize = 0,
  featureSize=0)
```

Arguments

- **size**: The number of candidate variables to be tested (the first `size` variables from `variableList`)
- **fraction**: The fraction of data (sampled with replacement) to be used as train
- **pvalue**: The maximum p-value, associated to the NeRI, allowed for a term in the model (controls the false selection rate)
- **loops**: The number of bootstrap loops
- **covariates**: A string of the type "1 + var1 + var2" that defines which variables will always be included in the models (as covariates)
- **Outcome**: The name of the column in `data` that stores the variable to be predicted by the model
- **variableList**: A data frame with two columns. The first one must have the names of the candidate variables and the other one the description of such variables
- **data**: A data frame where all variables are stored in different columns
- **maxTrainModelSize**: Maximum number of terms that can be included in the model
- **type**: Fit type: Logistic ("LOGIT"), linear ("LM"), or Cox proportional hazards ("COX")
- **testType**: Type of non-parametric test to be evaluated by the `improvedResiduals` function: Binomial test ("Binomial"), Wilcoxon rank-sum test ("Wilcoxon"), Student’s t-test ("tStudent"), or F-test ("Ftest")
- **timeOutcome**: The name of the column in `data` that stores the time to event (needed only for a Cox proportional hazards regression model fitting)
- **cores**: Cores to be used for parallel processing
- **randsize**: the model size of a random outcome. If randsize is less than zero. It will estimate the size
- **featureSize**: The original number of features to be explored in the data frame.
Value

final.model  An object of class `lm`, `glm`, or `coxph` containing the final model
var.names   A vector with the names of the features that were included in the final model
formula    An object of class `formula` with the formula used to fit the final model
ranked.var  An array with the ranked frequencies of the features
formula.list A list containing objects of class `formula` with the formulas used to fit the models found at each cycle
variableList A list of variables used in the forward selection

Author(s)

Jose G. Tamez-Pena and Antonio Martinez-Torteya

See Also

`forwardselectionModelBin`

Description

This function uses a wrapper procedure to select the best features of a non-penalized linear model that best predict the outcome, given the formula of an initial model template (linear, logistic, or Cox proportional hazards), an optimization procedure, and a data frame. A filter scheme may be enabled to reduce the search space of the wrapper procedure. The false selection rate may be empirically controlled by enabling bootstrapping, and model shrinkage can be evaluated by cross-validation.

Usage

```r
FRESA.Model(formula, data, 
  OptType = c("Binary", "Residual"), 
  pvalue = 0.05, 
  filter.p.value = 0.10, 
  loops = 32, 
  maxTrainModelSize = 20, 
  elimination.bootstrap.steps = 100, 
  bootstrap.steps = 100, 
  print = FALSE, 
  plots = FALSE, 
  CVfolds = 1, 
  repeats = 1, 
  nk = 0, 
  categorizationType = c("Raw",
```
"Categorical",
"ZCategorical",
"RawZCategorical",
"RawTail",
"RawZTail",
"Tail",
"RawRawIL",
cateGroups = c(0.1, 0.9),
raw.dataFrame = NULL,
var.description = NULL,
testType = c("zIDI",
        "zNRI",
        "Binomial",
        "Wilcoxon",
        "tStudent",
        "Ftest"),
lambda="lambda.1se",
equivalent=FALSE,
bswimsCycles=20,
usrFitFun=NULL
)

Arguments

formula An object of class formula with the formula to be fitted
data A data frame where all variables are stored in different columns
OptType Optimization type: Based on the integrated discrimination improvement (Binary) index for binary classification ("Binary"), or based on the net residual improvement (NeRI) index for linear regression ("Residual")
pvalue The maximum p-value, associated to the testType, allowed for a term in the model (it will control the false selection rate)
filter.p.value The maximum p-value, for a variable to be included to the feature selection procedure
loops The number of bootstrap loops for the forward selection procedure
maxTrainModelSize Maximum number of terms that can be included in the model
elimination.bootstrap.steps The number of bootstrap loops for the backwards elimination procedure
bootstrap.steps The number of bootstrap loops for the bootstrap validation procedure
print Logical. If TRUE, information will be displayed
plots Logical. If TRUE, plots are displayed
CVfolds The number of folds for the final cross-validation
repeats The number of times that the cross-validation procedure will be repeated
FRESA.Model

nk The number of neighbors used to generate a \( k \)-nearest neighbors (KNN) classification. If zero, \( k \) is set to the square root of the number of cases. If less than zero, it will not perform the KNN classification.

categorizationType How variables will be analyzed: As given in data ("Raw"); broken into the \( p \)-value categories given by cateGroups ("Categorical"); broken into the \( p \)-value categories given by cateGroups and weighted by the \( z \)-score ("ZCategorical"); broken into the \( p \)-value categories given by cateGroups, weighted by the \( z \)-score, plus the raw values ("RawZCategorical"); raw values, plus the tails ("RawTail"); or raw values, weighted by the \( z \)-score, plus the tails ("RawZTail").

cateGroups A vector of percentiles to be used for the categorization procedure.

raw.dataframe A data frame similar to data, but with unadjusted data, used to get the means and variances of the unadjusted data.

var.description A vector of the same length as the number of columns of data, containing a description of the variables.

testType For a Binary-based optimization, the type of index to be evaluated by the ImproveProb function (Hmisc package): \( z \)-value of Binary or of NRI. For a NeRI-based optimization, the type of non-parametric test to be evaluated by the improvedResiduals function: Binomial test ("Binomial"), Wilcoxon rank-sum test ("Wilcox"), Student’s \( t \)-test ("tStudent"), or \( F \)-test ("Ftest").

lambda The passed value to the \( s \) parameter of the glmnet cross validation coefficient equivalent Is set to TRUE CV will compute the equivalent model.

bswimsCycles The maximum number of models to be returned by BSWiMS.model.

usrFitFun An optional user provided fitting function to be evaluated by the cross validation procedure: fitting: usrFitFun(formula,data), with a predict function.

Details

This important function of FRESA.CAD will model or cross validate the models. Given an outcome formula, and a data.frame this function will do an univariate analysis of the data (univariateRankVariables), then it will select the top ranked variables; after that it will select the model that best describes the outcome. At output it will return the bootstrapped performance of the model (bootstrapValidation_Bin or bootstrapValidation_Res). It can be set to report the cross-validation performance of the selection process which will return either a crossValidationFeatureSelection_Bin or a crossValidationFeatureSelection_Res object.

Value

BSWiMS.model An object of class lm, glm, or coxph containing the final model.

reducedModel The resulting object of the backward elimination procedure.

univariateAnalysis A data frame with the results from the univariate analysis.

forwardModel The resulting object of the feature selection function.

updatedforwardModel The resulting object of the the update procedure.
FRESA.Model

bootstrappedModel The resulting object of the bootstrap procedure on final.model
cvObject The resulting object of the cross-validation procedure
used.variables The number of terms that passed the filter procedure
call the function call

Author(s)

Jose G. Tamez-Pena and Antonio Martinez-Torteya

References


Examples

```r
## Not run:

# Start the graphics device driver to save all plots in a pdf format
pdf(file = "FRESA.Model.Example.pdf", width = 8, height = 6)
# Get the stage C prostate cancer data from the rpart package
data(stagec.package = "rpart")
options(NA.action = "na.pass")
stagec_mat <- cbind(pgstat = stagec$pgstat,
                   pgt ime = stagec$pgtime,
                   as.data.frame(model.matrix(Surv(pgt ime,pgstat) ~ .,stagec))[-1])
data(cancerVarNames)
dataCancerImputed <- nearestNeighborImpute(stagec_mat)
# Get a Cox proportional hazards model using:  
# - The default parameters
md <- FRESA.Model(formula = Surv(pgt ime, pgstat) ~ 1,
                   data = dataCancerImputed,
                   var.description = cancerVarNames[,2])
pt <- plot(md$bootstrappedModel)
sm <- summary(md$BSW1MS.model)
print(sm$coefficients)

# Get a 10 fold CV Cox proportional hazards model using:  
# - Repeat 10 times de CV
md <- FRESA.Model(formula = Surv(pgt ime, pgstat) ~ 1,
                   data = dataCancerImputed, CVfolds = 10,
                   repeats = 10,
                   var.description = cancerVarNames[,2])
pt <- plotModels.ROC(md$cvObject$Models.testPrediction, theCVfolds = 10)
print(pt$predictionTable)
```
pt <- plotModels.ROC(md$cvObject$LASSO.testPredictions, theCVfolds = 10)
pt <- plotModels.ROC(md$cvObject$KNN.testPrediction, theCVfolds = 10)

# Get a regression of the survival time

timeSubjects <- dataCancerImputed
timeSubjects$pgtime <- log(timeSubjects$pgtime)

md <- FRESA.Model(formula = pgtime ~ 1,
                   data = timeSubjects,
                   var.description = cancerVarNames[,2])
pt <- plot(md$bootstrappedModel)
sm <- summary(md$BSWiMS.model)
print(sm$coefficients)

# Get a logistic regression model using
# - The default parameters and removing time as possible predictor

dataCancerImputed$pgtime <- NULL

md <- FRESA.Model(formula = pgstat ~ 1,
                   data = dataCancerImputed,
                   var.description = cancerVarNames[,2])
pt <- plot(md$bootstrappedModel)
sm <- summary(md$BSWiMS.model)
print(sm$coefficients)

# Get a logistic regression model using:
# - residual-based optimization
md <- FRESA.Model(formula = pgstat ~ 1,
                   data = dataCancerImputed,
                   OptType = "Residual",
                   var.description = cancerVarNames[,2])
pt <- plot(md$bootstrappedModel)
sm <- summary(md$BSWiMS.model)
print(sm$coefficients)

# Shut down the graphics device driver
dev.off()

## End(Not run)

---

FRESAScale  

Data frame normalization

Description

All features from the data will be normalized based on the distribution of the reference data-frame
Usage

FRESAScale(data, refFrame=NULL, method=c("Norm", "Order", "RankInv"), refMean=NULL, refDisp=NULL)

Arguments

data The dataframe to be normalized
refFrame The reference frame that will be used to extract the feature distribution
method The normalization method. Norm: Mean and Std, Order: Median and IQR, RankInv: rankInverseNormalDataFrame
refMean The mean vector of the reference frame
refDisp the data dispersion method of the reference frame

Details

The data-frame will be normalized according to the distribution of the reference frame or the mean vector(refMean) scaled by the reference dispersion vector(refDisp).

Value

scaledData The scaled data set
refMean The mean or median vector of the reference frame
refDisp The data dispersion (standard deviation or IQR)

Author(s)

Jose G. Tamez-Pena

See Also

rankInverseNormalDataFrame

________________________________________________________________

getKNNpredictionFromFormula

Predict classification using KNN

________________________________________________________________

Description

This function will return the classification of the samples of a test set using a k-nearest neighbors (KNN) algorithm with euclidean distances, given a formula and a train set.
Usage

```r
getKNNpredictionFromFormula(model.formula, 
   testData, 
   testData, 
   Outcome = "CLASS", 
   nk = 3)
```

Arguments

- **model.formula**: An object of class formula with the formula to be used
- **trainData**: A data frame with the data to train the model, where all variables are stored in different columns
- **testData**: A data frame similar to trainData, but with the data set to be predicted
- **Outcome**: The name of the column in trainData that stores the variable to be predicted by the model
- **nk**: The number of neighbors used to generate the KNN classification

Value

- **prediction**: A vector with the predicted outcome for the testData data set
- **prob**: The proportion of k neighbors that predicted the class to be the one being reported in prediction
- **binProb**: The proportion of k neighbors that predicted the class of the outcome to be equal to 1
- **featureList**: A vector with the names of the features used by the KNN procedure

Author(s)

Jose G. Tamez-Pena and Antonio Martinez-Torteya

See Also

- `predict.fitFRESA`, `knn`

Description

This function returns the matrix template [mean, sd, IQR] that maximizes the ROC AUC between cases of controls.
Usage

gSignature(
  data, 
  varlist=NULL, 
  Outcome=NULL, 
  target=c("All","Control","Case"), 
  CVFolds=3, 
  repeats=9, 
  distanceFunction=signatureDistance, 
  ... 
)

Arguments

data A data frame whose rows contain the sampled "subject" data, and each column
  is a feature.
varlist The varlist is a character vector that list all the features to be searched by the
  Backward elimination forward selection procedure.
Outcome The name of the column that has the binary outcome. 1 for cases, 0 for controls
  target The target template that will be used to maximize the AUC.
CVFolds The number of folds to be used
repeats how many times the CV procedure will be repeated
distanceFunction The function to be used to compute the distance between the template and each
  sample
... the parameters to be passed to the distance function

Details

The function repeats full cycles of a Cross Validation (RCV) procedure. At each CV cycle the
algorithm estimate the mean template and the distance between the template and the test samples.
The ROC AUC is computed after the RCV is completed. A forward selection scheme. The set of
features that maximize the AUC during the Forward loop is returned.

Value

cTemplate the control matrix with quantile probs[0.025,0.25,0.5,0.75,0.975] that maximized
  the AUC (template of controls subjects)
caseTTemplate the case matrix with quantile probs[0.025,0.25,0.5,0.75,0.975] that maximized
  the AUC (template of case subjects)
AUCevolution The AUC value at each cycle
featureSizeEvolution The number of features at each cycle
featureList The final list of features
CVOutput: A data frame with four columns: ID, Outcome, Case Distances, Control Distances. Each row contains the CV test results.

MaxAUC: The maximum ROC AUC.

**Author(s)**

Jose G. Tamez-Pena

---

`getVar.Bin` - Analysis of the effect of each term of a binary classification model by analysing its reclassification performance.

**Description**

This function provides an analysis of the effect of each model term by comparing the binary classification performance between the Full model and the model without each term. The model is fitted using the train data set, but probabilities are predicted for the train and test data sets. Reclassification improvement is evaluated using the `improveProb` function (`hmisc` package). Additionally, the integrated discrimination improvement (IDI) and the net reclassification improvement (NRI) of each model term are reported.

**Usage**

```
getVar.Bin(object, data, Outcome = "Class", type = c("LOGIT", "LM", "COX"), testData = NULL, callCpp=TRUE)
```

**Arguments**

- **object**: An object of class `lm`, `glm`, or `coxph` containing the model to be analysed.
- **data**: A data frame where all variables are stored in different columns.
- **Outcome**: The name of the column in `data` that stores the variable to be predicted by the model.
- **type**: Fit type: Logistic ("LOGIT"), linear ("LM"), or Cox proportional hazards ("COX").
- **testData**: A data frame similar to `data`, but with a data set to be independently tested. If NULL, `data` will be used.
- **callCpp**: is set to true it will use the c++ implementation of improvement.
**Value**

- **z.IDIs**: A vector in which each term represents the $z$-score of the IDI obtained with the Full model and the model without one term.
- **z.NRIs**: A vector in which each term represents the $z$-score of the NRI obtained with the Full model and the model without one term.
- **IDIs**: A vector in which each term represents the IDI obtained with the Full model and the model without one term.
- **NRIs**: A vector in which each term represents the NRI obtained with the Full model and the model without one term.
- **testData.z.IDIs**: A vector similar to z.IDIs, where values were estimated in testdata.
- **testData.z.NRIs**: A vector similar to z.NRIs, where values were estimated in testdata.
- **testData.IDIs**: A vector similar to IDIs, where values were estimated in testdata.
- **testData.NRIs**: A vector similar to NRIs, where values were estimated in testdata.
- **uniTrainAccuracy**: A vector with the univariate train accuracy of each model variable.
- **uniTestAccuracy**: A vector with the univariate test accuracy of each model variable.

**Author(s)**

Jose G. Tamez-Pena and Antonio Martinez-Torteya

**References**


**See Also**

- **getVar.Res**

**Description**

This function provides an analysis of the effect of each model term by comparing the residuals of the Full model and the model without each term. The model is fitted using the train data set, but analysis of residual improvement is done on the train and test data sets. Residuals are compared by a paired $t$-test, a paired Wilcoxon rank-sum test, a binomial sign test and the $F$-test on residual variance. Additionally, the net residual improvement (NeRI) of each model term is reported.
Usage

getVar.Res(object, data, Outcome = "Class", type = c("LM", "LOGIT", "COX"), testData = NULL, callCpp=TRUE)

Arguments

object An object of class lm, glm, or coxph containing the model to be analyzed
data A data frame where all variables are stored in different columns
Outcome The name of the column in data that stores the variable to be predicted by the model
type Fit type: Logistic ("LOGIT"), linear ("LM"), or Cox proportional hazards ("COX")
testData A data frame similar to data, but with a data set to be independently tested. If NULL, data will be used.
callCpp is set to true it will use the c++ implementation of residual improvement.

Value

tP.value A vector in which each element represents the single sided p-value of the paired t-test comparing the absolute values of the residuals obtained with the Full model and the model without one term
BinP.value A vector in which each element represents the p-value associated with a significant improvement in residuals according to the binomial sign test
WilcoxF.value A vector in which each element represents the single sided p-value of the Wilcoxon rank-sum test comparing the absolute values of the residuals obtained with the Full model and the model without one term
FP.value A vector in which each element represents the single sided p-value of the F-test comparing the residual variances of the residuals obtained with the Full model and the model without one term
NeRIs A vector in which each element represents the net residual improvement between the Full model and the model without one term
testData.tP.value A vector similar to tP.value, where values were estimated in testData
testData.BinP.value A vector similar to BinP.value, where values were estimated in testData
testData.WilcoxF.value A vector similar to WilcoxF.value, where values were estimated in testData
testData.FP.value A vector similar to FP.value, where values were estimated in testData
testData.NeRIs A vector similar to NeRIs, where values were estimated in testData
unittestMSE A vector with the univariate residual mean sum of squares of each model variable on the test data
unitrainMSE  A vector with the univariate residual mean sum of squares of each model variable on the train data

Author(s)
Jose G. Tamez-Pena and Antonio Martinez-Torteya

See Also
getVar.Bin

heatMaps  Plot a heat map of selected variables

Description
This function creates a heat map for a data set based on a univariate or frequency ranking

Usage
heatMaps(variableList=NULL, varRank = NULL, Outcome, data, title = "Heat Map", hCluster = FALSE, prediction = NULL, Scale = FALSE, theFiveColors=c("blue","cyan","black","yellow","red"), outcomeColors = c("blue","lightgreen","yellow","orangered","red"), transpose=FALSE, ...)  

Arguments
variableList  A data frame with two columns. The first one must have the names of the candidate variables and the other one the description of such variables

varRank  A data frame with the name of the variables in variableList, ranked according to a certain metric

Outcome  The name of the column in data that stores the variable to be predicted by the model

data  A data frame where all variables are stored in different columns

title  The title of the plot

hCluster  Logical. If TRUE, variables will be clustered

prediction  A vector with a prediction for each subject, which will be used to rank the heat map
Scale: An optional value to force the data normalization outcome

theFiveColors: the colors of the heatmap

outcomeColors: the colors of the outcome bar

transpose: transpose the heatmap
...

additional parameters for the heatmap.2 function

Value

dataMatrix: A matrix with all the terms in data described by variableList

orderMatrix: A matrix similar to dataMatrix, where rows are ordered according to the outcome

heatMap: A list with the values returned by the heatmap.2 function (gplots package)

Author(s)

Jose G. Tamez-Pena and Antonio Martinez-Torteya

Examples

```r
## Not run:

library(rpart)
data(stagec)

# Set the options to keep the na
options(na.action='na.pass')
# create a model matrix with all the NA values imputed
stagecImputed <- as.data.frame(nearestNeighborImpute(model.matrix(~., stagec)[,-1]))

# the simple heat map
hm <- heatmap(outcome="pgstat", data=stagecImputed, title="Heat Map", scale=TRUE)

# transposing the heat-map with clustered columns
hm <- heatmap(outcome="pgstat", data=stagecImputed, title="Heat Map", scale=TRUE, transpose=TRUE, hCluster = TRUE, 
              cexRow=0.80, cexCol=0.50, srtCol=35)

# transposing the heat-map with reds and time to event as outcome
hm <- heatmap(outcome="pgtime", data=stagecImputed, title="Heat Map", scale=TRUE, 
              theFiveColors=c("black", "red", "orange", "yellow", "white"), 
              cexRow=0.50, cexCol=0.80, srtCol=35)
```

## End(Not run)
improvedResiduals

Estimate the significance of the reduction of predicted residuals

**Description**

This function will test the hypothesis that, given a set of two residuals (new vs. old), the new ones are better than the old ones as measured with non-parametric tests. Four p-values are provided: one for the binomial sign test, one for the paired Wilcoxon rank-sum test, one for the paired t-test, and one for the F-test. The proportion of subjects that improved their residuals, the proportion that worsen their residuals, and the net residual improvement (NeRI) will be returned.

**Usage**

```r
improvedResiduals(oldResiduals, newResiduals, testType = c("Binomial", "Wilcox", "tStudent", "Ftest"))
```

**Arguments**

- `oldResiduals`: A vector with the residuals of the original model
- `newResiduals`: A vector with the residuals of the new model
- `testType`: Type of non-parametric test to be evaluated: Binomial test ("Binomial"), Wilcoxon rank-sum test ("Wilcox"), Student’s t-test ("tStudent"), or F-test ("Ftest")

**Details**

This function will test the hypothesis that the new residuals are "better" than the old residuals. To test this hypothesis, four types of tests are performed:

1. The paired t-test, which compares the absolute value of the residuals
2. The paired Wilcoxon rank-sum test, which compares the absolute value of residuals
3. The binomial sign test, which evaluates whether the number of subjects with improved residuals is greater than the number of subjects with worsened residuals
4. The F-test, which is the standard test for evaluating whether the residual variance is "better" in the new residuals.

The proportions of subjects that improved and worsen their residuals are returned, and so is the NeRI.

**Value**

- `p1`: Proportion of subjects that improved their residuals to the total number of subjects
- `p2`: Proportion of subjects that worsen their residuals to the total number of subjects
- `NeRI`: The net residual improvement (p1-p2)
- `p.value`: The one tail p-value of the test specified in `testType`
KNN_method

BinP.value  The p-value associated with a significant improvement in residuals
WilcoxFP.value The single sided p-value of the Wilcoxon rank-sum test comparing the absolute values of the new and old residuals
tP.value The single sided p-value of the paired t-test comparing the absolute values of the new and old residuals
FP.value The single sided p-value of the F-test comparing the residual variances of the new and old residuals

Author(s)
Jose G. Tamez-Pena and Antonio Martinez-Torteya

KNN_method

KNN Setup for KNN prediction

Description
Prepares the KNN function to be used to predict the class of a new set

Usage
KNN_method(formula = formula, data=NULL,...)

Arguments
formula the base formula to extract the outcome
data the data to be used for training the KNN method
... parameters for the KNN function and the data scaling method

Value
trainData The data frame to be used to train the KNN prediction
scaledData The scaled training set
classData A vector with the outcome to be used by the KNN function
outcome The name of the outcome
usedFeatures The list of features to be used by the KNN method
mean_col A vector with the mean of each training feature
disp_col A vector with the dispersion of each training feature
kn The number of neighbors to be used by the predict function
scaleMethod The scaling method to be used by FRESAScale() function

Author(s)
Jose G. Tamez-Pena
See Also

class::knn,FRESAScale

---

### LASSO

CV LASSO fit with s="lambda.min" or s="lambda.1se"

---

### Description

Fits a glmnet::cv.glmnet object to the data, and sets the prediction to use the features that created the minimum CV error or one SE.

### Usage

```
LASSO_MIN(formula = formula, data=NULL,...)  
LASSO_1SE(formula = formula, data=NULL,...)
```

### Arguments

- `formula` The base formula to extract the outcome
- `data` The data to be used for training the KNN method
- `...` Parameters to be passed to the cv.glmnet function

### Value

- `fit` The glmnet::cv.glmnet fitted object
- `s` The s. Set to "lambda.min" or "lambda.1se" for prediction
- `formula` The formula
- `outcome` The name of the outcome
- `usedFeatures` The list of features to be used

### Author(s)

Jose G. Tamez-Pena

### See Also

`glmnet::cv.glmnet`
**listTopCorrelatedVariables**

*List the variables that are highly correlated with each other*

**Description**

This function computes the Pearson, Spearman, or Kendall correlation for each specified variable in the data set and returns a list of the variables that are correlated to them. It also provides a short variable list without the highly correlated variables.

**Usage**

```r
listTopCorrelatedVariables(variableList, data, pvalue = 0.001, corthreshold = 0.9, method = c("pearson", "kendall", "spearman"))
```

**Arguments**

- `variableList`: A data frame with two columns. The first one must have the names of the candidate variables and the other one the description of such variables.
- `data`: A data frame where all variables are stored in different columns.
- `pvalue`: The maximum $p$-value, associated to `method`, allowed for a pair of variables to be defined as significantly correlated.
- `corthreshold`: The minimum correlation score, associated to `method`, allowed for a pair of variables to be defined as significantly correlated.
- `method`: Correlation method: Pearson product-moment ("pearson"), Spearman’s rank ("spearman"), or Kendall rank ("kendall").

**Value**

- `correlated.variables`: A data frame with two columns:
  1. cor.var.names: The variables that are correlated
  2. cor.var.value: The correlation value
- `short.list`: A vector with a list of variables that are not correlated to each other. For every correlated pair, only the variable that first entered the correlation analysis was kept.

**Author(s)**

Jose G. Tamez-Pena and Antonio Martinez-Torteya
Examples

```r
## Not run:
# Start the graphics device driver to save all plots in a pdf format
pdf(file = "Example.pdf")
# Get the stage C prostate cancer data from the rpart package
library(rpart)
data(stagec)
# Split the stages into several columns
dataCancer <- cbind(stagec[c(1:3,5:6)],
    gleason4 = 1*(stagec[,7] == 4),
    gleason5 = 1*(stagec[,7] == 5),
    gleason6 = 1*(stagec[,7] == 6),
    gleason7 = 1*(stagec[,7] == 7),
    gleason8 = 1*(stagec[,7] == 8),
    gleason910 = 1*(stagec[,7] >= 9),
    eet = 1*(stagec[,4] == 2),
    diploid = 1*(stagec[,8] == "diploid"),
    tetraploid = 1*(stagec[,8] == "tetraploid"),
    notAneuploid = 1-1*(stagec[,8] == "aneuploid")
)
# Remove the incomplete cases
dataCancer <- dataCancer[complete.cases(dataCancer),]
# Load a pre-established data frame with the names and descriptions of all variables
data(cancervarnames)
# Get the variables that have a correlation coefficient larger
# than 0.65 at a p-value of 0.05
cor <- listTopCorrelatedVariables(variableList = cancervarnames,
    data = dataCancer,
    pvalue = 0.05,
    corthreshold = 0.65,
    method = "pearson")
# Shut down the graphics device driver
dev.off()
## End(Not run)
```

**LM_RIDGE_MIN**  
*Ridge Linear Models*

**Description**

FRESA wrapper to fit MASS::lm.ridge object to the data and returning the coef with minimum GCV

**Usage**

`LM_RIDGE_MIN(formula = formula, data=NULL,...)`
modelFitting

Arguments

- `formula` The base formula to extract the outcome
- `data` The data to be used for training the method
- `...` Parameters to be passed to the MASS::lm.ridge function

Value

- `fit` The MASS::lm.ridge fitted object

Author(s)

Jose G. Tamez-Pena

See Also

MASS::lm.ridge

---

### Description

This function fits a linear, logistic, or Cox proportional hazards regression model to given data.

### Usage

```r
modelFitting(model.formula, 
  data, 
  type = c("LOGIT", "LM", "COX","SVM"), 
  fitFRESA=TRUE, 
  ...)
```

### Arguments

- `model.formula` An object of class `formula` with the formula to be used
- `data` A data frame where all variables are stored in different columns
- `type` Fit type: Logistic ("LOGIT"), linear ("LM"), Cox proportional hazards ("COX") or "SVM"
- `fitFRESA` if true it will perform use the FRESA cpp code for fitting
- `...` Additional parameters for fitting a default glm object

### Value

A fitted model of the type defined in `type`

### Author(s)

Jose G. Tamez-Pena and Antonio Martinez-Torteya
mRMR.classic_FRESA  
FRESA.CAD wrapper of mRMR::mRMR.classic

Description
Returns the positive MI-scored set of maximum relevance minimum redundancy (mRMR) features returned by the mMRM.classic function

Usage
mRMR.classic_FRESA(data=NULL, Outcome=NULL, feature_count=0,...)

Arguments
- data: The data frame
- Outcome: The outcome feature
- feature_count: The number of features to return
- ...: Extra parameters to be passed to the mRMR::mRMR.classic function

Value
Named vector with the MI-score of the selected features

Author(s)
Jose G. Tamez-Pena

See Also
mRMR::mRMR.classic

---

NAIVE_BAYES  
Naive Bayes Modeling

Description
FRESA wrapper to fit naivebayes::naive_bayes object to the data

Usage
NAIVE_BAYES(formula = formula, data=NULL,...)
Arguments

- **formula**: The base formula to extract the outcome
- **data**: The data to be used for training the method
- ... Parameters to be passed to the naivebayes::naive_bayes function

Value

- **fit**: The naivebayes::naive_bayes fitted object

Author(s)

Jose G. Tamez-Pena

See Also

naivebayes::naive_bayes

Description

The function will replace any NA present in the data-frame with the median values of the nearest neighbours.

Usage

```r
earestNeighborImpute(tobeimputed, 
  referenceSet=NULL, 
  distol=1.05 
)
```

Arguments

- **tobeimputed**: a data frame with missing values (NA values)
- **referenceSet**: An optional data frame with a set of complete observations. This data frame will be added to the search set
- **distol**: The tolerance used to define if a particular set of row observations is similar to the minimum distance

Details

This function will find any NA present in the data set and it will search for the row set of complete observations that have the closest IQR normalized Manhattan distance to the row with missing values. If a set of rows have similar minimum distances (toldis*(minimum distance) > row set distance) the median value will be used.
Value

A data frame, where each NA has been replaced with the value of the nearest neighbors

Author(s)

Jose G. Tamez-Pena

Examples

```r
## Not run:
# Get the stage C prostate cancer data from the rpart package
library(rpart)
data(stagec)
# Set the options to keep the na
options(na.action='na.pass')
# create a model matrix with all the NA values imputed
stagecimputed <- nearestNeighborImpute(model.matrix(~ ., stagec), [-1])

## End(Not run)
```

---

**plot.bootstrapValidation_Bin**

*Plot ROC curves of bootstrap results*

Description

This function plots ROC curves and a Kaplan-Meier curve (when fitting a Cox proportional hazards regression model) of a bootstrapped model.

Usage

```r
## S3 method for class 'bootstrapValidation_Bin'
plot(x, 
  xlab = "Years",
  ylab = "Survival",
  strata.levels=c(0),
  main = "ROC",
  cex=1.0,
  ...
)
```

Arguments

- **x** A bootstrapValidation_Bin object
- **xlab** The label of the x-axis
- **ylab** The label of the y-axis
- **strata.levels** Stratification level for the Kaplan-Meier plots
- **main** Main Plot title
The text cex

... Additional parameters for the generic plot function

Author(s)
José G. Tamez-Peña and Antonio Martínez-Torteya

See Also
plot.bootstrapValidation_Res

plot.bootstrapValidation_Res

Plot ROC curves of bootstrap results

Description
This function plots ROC curves and a Kaplan-Meier curve (when fitting a Cox proportional hazards regression model) of a bootstrapped model.

Usage
```r
# S3 method for class 'bootstrapValidation_Res'
plot(x,
     xlab = "Years",
     ylab = "Survival",
     ...)
```

Arguments
- x: A bootstrapValidation_Res object
- xlab: The label of the x-axis
- ylab: The label of the y-axis
- ...: Additional parameters for the plot

Author(s)
José G. Tamez-Peña and Antonio Martínez-Torteya

See Also
plot.bootstrapValidation_Bin
plot.FRESA_benchmark  

Plot the results of the model selection benchmark

Description

The different output metrics of the benchmark (BinaryBenchmark, RegresionBenchmark or OrdinalBenchmark) are plotted. It returns data matrices that describe the different plots.

Usage

```r
## S3 method for class 'FRESA_benchmark'
plot(x,...)
```

Arguments

- `x`: A FRESA_benchmark object
- `...`: Additional parameters for the generic `plot` function

Value

- `metrics`: The model test performance based on the `predictionStats_binary`, `predictionStats_regression` or `predictionStats_ordinal` functions.
- `barPlotsCI`: The `barPlotCiError` outputs for each metric.
- `metrics_filter`: The model test performance for each filter method based on the `predictionStats_binary` function.
- `barPlotsCI_filter`: The `barPlotCiError` outputs for each metric on the filter methods.
- `minMaxMetrics`: Reports the min and maximum value for each reported metric.

Author(s)

Jose G. Tamez-Pena

See Also

- `BinaryBenchmark`, `predictionStats_binary`
plotModels.ROC  

*Plot test ROC curves of each cross-validation model*

**Description**

This function plots test ROC curves of each model found in the cross validation process. It will also aggregate the models into a single prediction performance, plotting the resulting ROC curve (models coherence). Furthermore, it will plot the mean sensitivity for a given set of specificities.

**Usage**

```
plotModels.ROC(modelPredictions, number.of.models = 0, specificities = c(0.975, 0.95, 0.90, 0.80, 0.70, 0.60, 0.50, 0.40, 0.30, 0.20, 0.10, 0.05), theCVfolds = 1, predictor = "Prediction", cex = 1.0, ...
```

**Arguments**

- `modelPredictions`: A data frame returned by the `crossValidationFeatureSelection_Bin` function, either the `Models.testPrediction`, the `FullBSwMS.testPrediction`, the `Models.CVtestPredictions`, the `TestRetrained.blindPredictions`, the `KNN.testPrediction`, or the `LASSO.testPredictions` value
- `number.of.models`: The maximum number of models to plot
- `specificities`: Vector containing the specificities at which the ROC sensitivities will be calculated
- `theCVfolds`: The number of folds performed in a Cross-validation experiment
- `predictor`: The name of the column to be plotted
- `cex`: Controlling the font size of the text inside the plots
- `...`: Additional parameters for the `roc` function (pROC package)

**Value**

- `ROC.AUCs`: A vector with the AUC of each ROC
- `mean.sensitivities`: A vector with the mean sensitivity at the specificities given by `specificities`
- `model.sensitivities`: A matrix where each row represents the sensitivity at the specificity given by `specificities` for a different ROC
- `specificities`: The specificities used to calculate the sensitivities
- `senaUC`: The AUC of the ROC curve that resulted from using `mean.sensitivities`
**predict.fitFRESA**

- **predictionTable**
  - The confusion matrix between the outcome and the ensemble prediction
- **ensemblePrediction**
  - The ensemble (median prediction) of the repeated predictions

**Author(s)**

Jose G. Tamez-Pena and Antonio Martinez-Torteya

---

**Description**

This function returns the predicted outcome of a specific model. The model is used to generate linear predictions. The probabilistic values are generated using the logistic transformation on the linear predictors.

**Usage**

```r
## S3 method for class 'fitFRESA'
predict(object, 
    ...)  
```

**Arguments**

- **object**  
  - An object of class fitFRESA containing the model to be analyzed
- **...**  
  - A list with: testdata=testdata;predictType=c("linear","prob") and impute=FALSE. If impute is set to TRUE it will use the object model to impute missing data

**Value**

A vector with the predicted values

**Author(s)**

Jose G. Tamez-Pena and Antonio Martinez-Torteya

**See Also**

- `nearestNeighborImpute`
predict.FRESAKNN  

*Predicts class::knn models*

### Description

This function predicts the outcome from a FRESAKNN model

### Usage

```r
## S3 method for class 'FRESAKNN'
predict(object, ...)
```

### Arguments

- `object`  
  An object of class FRESAKNN containing the KNN train set
- `...`  
  A list with: testdata=testdata

### Value

A vector of the predicted values

### Author(s)

Jose G. Tamez-Pena

### See Also

KNN_method, class::knn

---

predict.FRESAsignature  

*Predicts CVsignature models*

### Description

This function predicts the outcome from a FRESAsignature model

### Usage

```r
## S3 method for class 'FRESAsignature'
predict(object, ...)
```

### Arguments

- `object`  
  An object of class FRESAsignature
- `...`  
  A list with: testdata=testdata
### predict.FRESA_LASSO

**Value**

A vector of the predicted values

**Author(s)**

Jose G. Tamez-Pena

**See Also**

CVsignature, getSignature, signatureDistance

---

The function predicts the outcome from a FRESA_LASSO fitted object.

**Usage**

```r
## S3 method for class 'FRESA_LASSO'
predict(object, ...)
```

**Arguments**

- `object`: An object of class FRESA_LASSO containing the model to be analyzed
- `...`: A list with: testdata=testdata

**Value**

A vector of the predicted values

**Author(s)**

Jose G. Tamez-Pena

**See Also**

LASSO
predict.FRESA_RIDGE

Predicts NAIVE_BAYES models

Description
This function predicts the outcome from a FRESA_NAIVEBAYES model

Usage
```r
## S3 method for class 'FRESA_NAIVEBAYES'
predict(object,...)
```

Arguments
- `object`  An object of class FRESA_NAIVEBAYES
- `...`     A list with: testdata=testdata

Value
A vector of the predicted values

Author(s)
Jose G. Tamez-Pena

See Also
- NAIVE_BAYES

predict.FRESA_RIDGE  Predicts LM_RIDGE_MIN models

Description
This function predicts the outcome from a LM_RIDGE_MIN model

Usage
```r
## S3 method for class 'FRESA_RIDGE'
predict(object,...)
```

Arguments
- `object`  An object of class FRESA_RIDGE
- `...`     A list with: testdata=testdata
Value

A vector of the predicted values

Author(s)

Jose G. Tamez-Pena

See Also

LM_RIDGE_MIN

predictionStats  Prediction Evaluation

Description

This function returns the statistical metrics describing the association between model predictions and the ground truth outcome.

Usage

predictionStats_binary(predictions, plotname="", center=FALSE,...)
predictionStats_regression(predictions, plotname="",...)
predictionStats_ordinal(predictions,plotname="",...)

Arguments

predictions  A matrix whose first column is the ground truth, and the second is the model prediction
plotname     The main title to be used by the plot function. If empty, no plot will be provided
center       For binary predictions indicates if the prediction is around zero
...          Extra parameters to be passed to the plot function.

Details

These functions will analyze the prediction outputs and will compare to the ground truth. The output will depend on the prediction task: Binary classification, Linear Regression or Ordinal regression.
**Value**

- **accc**: The classification accuracy with its 95% confidence intervals (95/CI).
- **berror**: The balanced error rate with its 95% CI.
- **aucs**: The ROC area under the curve (ROC AUC) of the binary classifier with its 95% CI.
- **specificity**: The specificity with its 95% CI.
- **sensitivity**: The sensitivity with its 95% CI.
- **ROC.analysis**: The output of the ROC function.
- **CM.analysis**: The output of the `epi::epi.tests` function.
- **corci**: The Pearson correlation with its 95% CI.
- **biasci**: The regression bias and its 95% CI.
- **RMSEci**: The root mean square error (RMSE) and its 95% CI.
- **spearman.ci**: The Spearman correlation and its 95% CI.
- **MAEci**: The mean absolute difference (MAE) and its 95% CI.
- **pearson**: The output of the `cor.test` function.
- **Kendall**: The Kendall correlation and its 95% CI.
- **Bias**: The ordinal regression bias and its 95% CI.
- **BMAE**: The balanced mean absolute difference for ordinal regression.
- **class95ci**: The output of the bootstrapped estimation of accuracy, sensitivity, and ROC AUC.
- **KendallTauB**: The output of the `DescTools::KendallTauB` function.
- **Kappa.analysis**: The output of the `irr::kappa2` function.

**Author(s)**

Jose G. Tamez-Pena

**See Also**

- `randomCV`
Usage

```r
randomCV(theData = NULL,
    theOutcome = "Class",
    fittingFunction=NULL,
    trainFraction = 0.5,
    repetitions = 100,
    trainSampleSets=NULL,
    featureSelectionFunction=NULL,
    featureSelection.control=NULL,
    asFactor=FALSE,
    addNoise=FALSE,
    ...
)
```

Arguments

- **theData**: The data-frame for cross-validation
- **theOutcome**: The name of the outcome
- **fittingFunction**: The fitting function used to model the data
- **trainFraction**: The percentage of the data to be used for training
- **repetitions**: The number of times that the CV process will be repeated
- **trainSampleSets**: A set of train samples
- **featureSelectionFunction**: The feature selection function to be used to filter out irrelevant features
- **featureSelection.control**: The parameters to control the feature selection function
- **asFactor**: Set theOutcome as factor
- **addNoise**: if TRUE will add 0.1
- **...**: Parameters to be passed to the fitting function

Value

- **testPredictions**: All the predicted outcomes. Is a data matrix with three columns c("Outcome","Model","Prediction"). Each row has a prediction for a given test subject
- **trainPredictions**: All the predicted outcomes in the train data set. Is a data matrix with three columns c("Outcome","Model","Prediction"). Each row has a prediction for a given test subject
- **medianTest**: The median of the test prediction for each subject
- **medianTrain**: The median of the prediction for each train subject
- **boxstaTest**: The statistics of the boxplot for test data
- **boxstaTrain**: The statistics of the boxplot for train data
trainSamplesSets
The id of the subjects used for training

selectedFeaturesSet
A list with all the features used at each training cycle

featureFrequency
A order table object that describes how many times a feature was selected.

jaccard
The jaccard index of the features as well as the average number of features used for prediction

theTimes
The CPU time analysis

Author(s)
José G. Tamez-Pena

Examples

```r
## Not run:

### Cross Validation Example ###
# Start the graphics device driver to save all plots in a pdf format
pdf(file = "CrossValidationExample.pdf", width = 8, height = 6)

# Get the stage C prostate cancer data from the rpart package
data(stagec, package = "rpart")

# Prepare the data. Create a model matrix with interactions but no event time
stagec$ptime <- NULL
stagec$et <- as.factor(stagec$et)
options(na.action = 'na.pass')
stagec_mat <- cbind(pgstat = stagec$pgstat,
                     as.data.frame(model.matrix(pgstat ~ ., stagec))[-1])
fnames <- colnames(stagec_mat)
fnames <- str_replace_all(fnames, ":", "-")
colnames(stagec_mat) <- fnames

# Impute the missing data
dataCancerImputed <- nearestNeighborImpute(stagec_mat)
dataCancerImputed[,1:ncol(dataCancerImputed)] <- sapply(dataCancerImputed, as.numeric)

# Cross validating a Random Forest classifier
cvRF <- randomCV(dataCancerImputed,"pgstat",
                  randomForest::randomForest,
                  trainFraction = 0.8,
                  repetitions = 10,
                  asFactor = TRUE);

# Evaluate the prediction performance of the Random Forest classifier
RFStats <- predictionStats_binary(cvRF$medianTest,
                                   plotname = "Random Forest", cex = 0.9);

# Cross validating a BSWiMS with the same train/test set
```
cvBSwims <- randomCV(fittingFunction = BSWims.model, 
  trainSampleSets = cvRF$trainSamplesSets);

# Evaluate the prediction performance of the BSWims classifier 
BSWimsStats <- predictionStats_binary(cvBSwims$medianTest, 
  plotname = "BSWims", cex = 0.9);

# Cross validating a LDA classifier with a t-student filter 
CVLDA <- randomCV(dataCancerImputed, pgstat, MASS::lda, 
  trainSampleSets = cvRF$trainSamplesSets, 
  featureSelectionFunction = univariate_tstudent, 
  featureSelection.control = list(limit = 0.5, thr = 0.975));

# Evaluate the prediction performance of the LDA classifier 
LDAStats <- predictionStats_binary(cvLDA$medianTest, plotname = "LDA", cex = 0.9);

# Cross validating a QDA classifier with LDA t-student features and RF train/test set 
CVQDA <- randomCV(fittingFunction = MASS::qda, 
  trainSampleSets = cvRF$trainSamplesSets, 
  featureSelectionFunction = cvLDA$selectedFeaturesSet);

# Evaluate the prediction performance of the QDA classifier 
QDAStats <- predictionStats_binary(cvQDA$medianTest, plotname = "QDA", cex = 0.9);

# Create a barplot with 95 errorcitable <- rbind(RFStats$error, 
  BSWimsStats$error, 
  LDAStats$error, 
  QDAStats$error)

bpCI <- barPlotCiError(as.matrix(errorcitable), metricname = "Balanced Error", 
  thesets = c("Classifier Method"), 
  themethod = c("RF","BSWims","LDA","QDA"), 
  main = "Balanced Error", 
  offsets = c(0.5,0.15), 
  scoreDirection = "<", 
  ho = 0.5, 
  args.legend = list(bg = "white", x = "topright"), 
  col = terrain.colors(4));

dev.off()

## End(Not run)
Description

This function takes a data frame and a reference control population to return a \( z \)-transformed data set conditioned to the reference population. Each sample data for each feature column in the data frame is conditionally \( z \)-transformed using a rank-based inverse normal transformation, based on the rank of the sample in the reference frame.

Usage

\[
\text{rankInverseNormalDataFrame}(\text{variableList, data, referenceframe, strata=NA})
\]

Arguments

- \text{variableList} A data frame with two columns. The first one must have the names of the candidate variables and the other one the description of such variables.
- \text{data} A data frame where all variables are stored in different columns.
- \text{referenceframe} A data frame similar to \text{data}, but with only the control population.
- \text{strata} The name of the column in \text{data} that stores the variable that will be used to stratify the model.

Value

A data frame where each observation has been conditionally \( z \)-transformed, given control data.

Author(s)

Jose G. Tamez-Pena and Antonio Martinez-Torteya

Examples

```r
## Not run:
# Start the graphics device driver to save all plots in a pdf format
pdf(file = "Example.pdf")
# Get the stage C prostate cancer data from the rpart package
library(rpart)
data(stagec)
# Split the stages into several columns
dataCancer <- cbind(stagec[,1:3:5:6],
geason4 = 1*(stagec[,7] == 4),
geason5 = 1*(stagec[,7] == 5),
geason6 = 1*(stagec[,7] == 6),
geason7 = 1*(stagec[,7] == 7),
geason8 = 1*(stagec[,7] == 8),
geason9 = 1*(stagec[,7] >= 9),
eet = 1*(stagec[,4] == 2),
diploid = 1*(stagec[,8] == "diploid"),
tetraploid = 1*(stagec[,8] == "tetraploid"),
notAneuploid = 1-1*(stagec[,8] == "aneuploid")
```

```r
```
# Remove the incomplete cases
dataCancer <- dataCancer[complete.cases(dataCancer),]

# Load a pre-established data frame with the names and descriptions of all variables
data(cancerVarNames)

# Set the group of no progression
noProgress <- subset(dataCancer, pgstat==0)

# z-transform g2 values using the no-progression group as reference
dataCancerZTransform <- rankInverseNormalDataFrame(variableList = cancerVarNames[2,],
                                                  data = dataCancer,
                                                  referenceFrame = noProgress)

# Shut down the graphics device driver
dev.off()

## End(Not run)

---

reportEquivalentVariables

*Report the set of variables that will perform an equivalent IDI discriminant function*

### Description

Given a model, this function will report a data frame with all the variables that may be interchanged in the model without affecting its classification performance. For each variable in the model, this function will loop all candidate variables and report all of which result in an equivalent or better zIDI than the original model.

#### Usage

```r
reportEquivalentVariables(object, pvalue = 0.05, data, variableList, Outcome = "Class", timeOutcome=NULL, type = c("LOGIT", "LM", "COX"), description = ".", method="BH", osize=0, fitFRESA=TRUE)
```

#### Arguments

- **object**
  - An object of class `lm`, `glm`, or `coxph` containing the model to be analyzed

- **pvalue**
  - The maximum p-value, associated to the IDI, allowed for a pair of variables to be considered equivalent

- **data**
  - A data frame where all variables are stored in different columns

- **variableList**
  - A data frame with two columns. The first one must have the names of the candidate variables and the other one the description of such variables
Residuals calculation

**Outcome**
The name of the column in data that stores the variable to be predicted by the model

**timeOutcome**
The name of the column in data that stores the time to event

**type**
Fit type: Logistic ("LOGIT"), linear ("LM"), or Cox proportional hazards ("COX")

**description**
The name of the column in variableList that stores the variable description

**method**
The method used by the p-value adjustment algorithm

**osize**
The number of features used for p-value adjustment

**fitFRESA**
if TRUE it will use the cpp based fitting method

**Value**

**pvalueList**
A list with all the unadjusted p-values of the equivalent features per model variable

**equivalentMatrix**
A data frame with three columns. The first column is the original variable of the model. The second column lists all variables that, if interchanged, will not statistically affect the performance of the model. The third column lists the corresponding z-scores of the IDI for each equivalent variable.

**formulaList**
a character vector with all the equivalent formulas

**equivalentModel**
a bagged model that used all the equivalent formulas. The model size is limited by the number of observations

**Author(s)**

Jose G. Tamez-Pena

---

**residualForFRESA**

*Return residuals from prediction*

**Description**

Given a model and a new data set, this function will return the residuals of the predicted values. When dealing with a Cox proportional hazards regression model, the function will return the Martingale residuals.

**Usage**

```r
residualForFRESA(object, testData, Outcome, eta = 0.05)
```
**signatureDistance**

### Arguments

- **object**: An object of class `lm`, `glm`, or `coxph` containing the model to be analyzed.
- **testData**: A data frame where all variables are stored in different columns, with the data set to be predicted.
- **outcome**: The name of the column in `data` that stores the variable to be predicted by the model.
- **eta**: The weight of the contribution of the Martingale residuals, or 1 - the weight of the contribution of the classification residuals (only needed if `object` is of class `coxph`).

### Value

A vector with the residuals (i.e. the differences between the predicted and the real outcome).

### Author(s)

Jose G. Tamez-Pena and Antonio Martinez-Torteya

---

**signatureDistance**

*Distance to the signature template*

### Description

This function returns a normalized distance to the signature template.

### Usage

```r
signatureDistance(
  template,
  data=NULL,
  method = c("pearson","spearman","kendall","RSS","MAN")
)
```

### Arguments

- **template**: A list with a template matrix of the signature described with quantiles = [0.025,0.25,0.5,0.75,0.975].
- **data**: A data frame that will be used to compute the distance.
- **method**: The distance method.

### Details

The distance to the template: "pearson","spearman" and "kendall" distances are computed using the correlation function i.e. 1-r. "RSS" distance is the root sum square distance "MAN" Manhattan. The standardized $L^1$ distance.
summary.bootstrapValidation_Bin

Generate a report of the results obtained using the bootstrapValidation_Bin function

Description
This function prints two tables describing the results of the bootstrap-based validation of binary classification models. The first table reports the accuracy, sensitivity, specificity and area under the ROC curve (AUC) of the train and test data set, along with their confidence intervals. The second table reports the model coefficients and their corresponding integrated discrimination improvement (IDI) and net reclassification improvement (NRI) values.

Usage
## S3 method for class 'bootstrapValidation_Bin'
summary(object,
  ...)  

Arguments

object An object of class bootstrapValidation_Bin

Additional parameters for the generic summary function

Value

performance A vector describing the results of the bootstrapping procedure

summary An object of class summary.lm, summary.glm, or summary.coxph containing a summary of the analyzed model

coef A matrix with the coefficients, IDI, NRI, and the 95% confidence intervals obtained via bootstrapping

performance.table A matrix with the tabulated results of the blind test accuracy, sensitivity, specificities, and area under the ROC curve

Author(s)
Jose G. Tamez-Pena and Antonio Martinez-Torteya
summary.fitFRESA

See Also

summaryReport

summary.fitFRESA  

Returns the summary of the fit

Description

Returns a summary of fitted model created by the modelFitting function with the fitFRESA parameter set to TRUE

Usage

## S3 method for class 'fitFRESA'
summary(object,
  type=c("Improvement","Residual"),
  ci=c(0.025,0.975),
  data=NULL,
  ...)

Arguments

object  
fitted model with the modelFitting function

type  
the type of coefficient estimation

ci  
lower and upper limit of the ci estimation

data  
the data to be used for 95

Value

da list with the analysis results.

Author(s)

Jose G. Tamez-Pena

See Also

modelFitting, bootstrapValidation_Bin, bootstrapValidation_Res
summaryReport

Report the univariate analysis, the cross-validation analysis and the correlation analysis

Description

This function takes the variables of the cross-validation analysis and extracts the results from the univariate and correlation analyses. Then, it prints the cross-validation results, the univariate analysis results, and the correlated variables. As output, it returns a list of each one of these results.

Usage

summaryReport(univariateObject,
summaryBootstrap,
listOfCorrelatedVariables = NULL,
digits = 2)

Arguments

univariateObject
A data frame that contains the results of the univariateRankVariables function

summaryBootstrap
A list that contains the results of the summary_bootstrapValidation_Bin function

listOfCorrelatedVariables
A matrix that contains the correlated.variables value from the results obtained with the listTopCorrelatedVariables function

digits
The number of significant digits to be used in the print function

Value

performance.table
A matrix with the tabulated results of the blind test accuracy, sensitivity, specificities, and area under the ROC curve

coefStats
A data frame that lists all the model features along with its univariate statistics and bootstrapped coefficients

cor.variables
A matrix that lists all the features that are correlated to the model variables

Author(s)

Jose G. Tamez-Pena and Antonio Martinez-Torteya

See Also

summary.bootstrapValidation_Bin
timeSerieAnalysis

Fit the listed time series variables to a given model

Description

This function plots the time evolution and does a longitudinal analysis of time dependent features. Features listed are fitted to the provided time model (mixed effect model) with a generalized least squares (GLS) procedure. As output, it returns the coefficients, standard errors, $t$-values, and corresponding $p$-values.

Usage

timeSerieAnalysis(variableList, 
    baseModel, 
    data, 
    timevar = "time", 
    contime = ".", 
    Outcome = ".", 
    ..., 
    description = ".", 
    Ptoshow = c(1), 
    plegend = c("p"), 
    timesign = "-", 
    catgo.names = c("Control", "Case") 
)

Arguments

variableList  A data frame with two columns. The first one must have the names of the candidate variables and the other one the description of such variables
baseModel  A string of the type "1 + var1 + var2" that defines the model to which variables will be fitted
data  A data frame where all variables are stored in different columns
timevar  The name of the column in data that stores the visit ID
contime  The name of the column in data that stores the continuous time (e.g. days or months) that has elapsed since the baseline visit
Outcome  The name of the column in data that stores an optional binary outcome that may be used to show the stratified analysis
description  The name of the column in variableList that stores the variable description
Ptoshow  Index of the $p$-values to be shown in the plot
plegend  Legend of the $p$-values to be shown in the plot	imesign  The direction of the arrow of time
catgo.names  The legends of the binary categories...

Additional parameters to be passed to the gls function
Details

This function will plot the evolution of the mean value of the listed variables with its corresponding error bars. Then, it will fit the data to the provided time model with a GLS procedure and it will plot the fitted values. If a binary variable was provided, the plots will contain the case and control data. As output, the function will return the model coefficients and their corresponding $t$-values, and the standard errors and their associated $p$-values.

Value

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>coef</td>
<td>A matrix with the coefficients of the GLS fitting</td>
</tr>
<tr>
<td>std.Errors</td>
<td>A matrix with the standardized error of each coefficient</td>
</tr>
<tr>
<td>t.values</td>
<td>A matrix with the $t$-value of each coefficient</td>
</tr>
<tr>
<td>p.values</td>
<td>A matrix with the $p$-value of each coefficient</td>
</tr>
<tr>
<td>sigmas</td>
<td>The root-mean-square error of the fitting</td>
</tr>
</tbody>
</table>

Author(s)

Jose G. Tamez-Pena and Antonio Martinez-Torteya

Description

This function reports the mean and standard deviation for each feature in a model, and ranks them according to a user-specified score. Additionally, it does a Kolmogorov-Smirnov (KS) test on the raw and $z$-standardized data. It also reports the raw and $z$-standardized $t$-test score, the $p$-value of the Wilcoxon rank-sum test, the integrated discrimination improvement (IDI), the net reclassification improvement (NRI), the net residual improvement (NeRI), and the area under the ROC curve (AUC). Furthermore, it reports the $z$-value of the variable significance on the fitted model. Besides reporting an ordered data frame, this function returns all arguments as values, so that the results can be updates with the `update.unirankvar` if needed.

Usage

```r
uniRankVar(variableList, formula, Outcome, data, categorizationType = c("Raw", "Categorical", "ZCategorical", "RawZCategorical", "RawTail", "RawZTail", "Tail"),

```
Arguments

variableList A data frame with two columns. The first one must have the names of the candidate variables and the other one the description of such variables

formula An object of class formula with the formula to be fitted

Outcome The name of the column in data that stores an optional binary outcome that may be used to show the stratified analysis

data A data frame where all variables are stored in different columns

categorizationType How variables will be analyzed: As given in data ("Raw"); broken into the p-value categories given by cateGroups ("Categorical"); broken into the p-value categories given by cateGroups, and weighted by the z-score ("ZCategorical"); broken into the p-value categories given bycateGroups, weighted by the z-score, plus the raw values ("RawZCategorical"); raw values, plus the tails ("RawTail"); or raw values, weighted by the z-score, plus the tails ("RawZTail")

type Fit type: Logistic ("LOGIT"), linear ("LM"), or Cox proportional hazards ("COX")

rankingTest Variables will be ranked based on: The z-score of the IDI ("zIDI"), the z-score of the NRI ("zNRI"), the IDI ("IDI"), the NRI ("NRI"), the NeRI ("NeRI"), the z-score of the model fit ("Ztest"), the AUC ("AUC"), the Somers’ rank correlation ("Cstat"), or the Kendall rank correlation ("Kendall")

cateGroups A vector of percentiles to be used for the categorization procedure

raw.dataFrame A data frame similar to data, but with unadjusted data, used to get the means and variances of the unadjusted data

description The name of the column in variableList that stores the variable description

uniType Type of univariate analysis: Binary classification ("Binary") or regression ("Regression")
uniRankVar

FullAnalysis: If FALSE it will only order the features according to its z-statistics of the linear model
acovariates: the list of covariates
timeOutcome: the name of the Time to event feature

Details
This function will create valid dummy categorical variables if, and only if, data has been z-standardized. The p-values provided in cateGroups will be converted to its corresponding z-score, which will then be used to create the categories. If non z-standardized data were to be used, the categorization analysis would return wrong results.

Value
orderframe: A sorted list of model variables stored in a data frame
variableList: The argument variableList
formula: The argument formula
Outcome: The argument Outcome
data: The argument data
categorizationType: The argument categorizationType
type: The argument type
rankingTest: The argument rankingTest
cateGroups: The argument cateGroups
raw.dataFrame: The argument raw.dataFrame
description: The argument description
uniType: The argument uniType

Author(s)
Jose G. Tamez-Pena and Antonio Martinez-Torteya

References

See Also
update.uniRankVar, univariateRankVariables
univariateRankVariables

Univariate analysis of features

Description

This function reports the mean and standard deviation for each feature in a model, and ranks them according to a user-specified score. Additionally, it does a Kolmogorov-Smirnov (KS) test on the raw and z-standardized data. It also reports the raw and z-standardized t-test score, the p-value of the Wilcoxon rank-sum test, the integrated discrimination improvement (IDI), the net reclassification improvement (NRI), the net residual improvement (NeRI), and the area under the ROC curve (AUC). Furthermore, it reports the z-value of the variable significance on the fitted model.

Usage

univariateRankVariables(variableList, formula, Outcome, data,
categorizationType = c("Raw",
  "Categorical",
  "ZCategorical",
  "RawZCategorical",
  "RawTail",
  "RawZTail",
  "Tail",
  "RawRaw"),
type = c("LOGIT", "LM", "COX"),
rankingTest = c("zIDI",
  "znRI",
  "IDI",
  "NRI",
  "NeRI",
  "Ztest",
  "AUC",
  "CStat",
  "Kendall"),
cateGroups = c(0.1, 0.9),
raw.dataFrame = NULL,
description = ".",
uniType = c("Binary","Regression"),
FullAnalysis=TRUE,
acovariates = NULL,
timeOutcome = NULL
)
Arguments

variableList A data frame with the candidate variables to be ranked
formula An object of class formula with the formula to be fitted
Outcome The name of the column in data that stores the variable to be predicted by the model
data A data frame where all variables are stored in different columns
categorizationType How variables will be analyzed: As given in data ("Raw"); broken into the p-value categories given by cateGroups ("Categorical"); broken into the p-value categories given by cateGroups, and weighted by the z-score ("ZCategorical"); broken into the p-value categories given by cateGroups, weighted by the z-score, plus the raw values ("RawZCategorical"); raw values, plus the tails ("RawTail"); or raw values, weighted by the z-score, plus the tails ("RawZTail")
type Fit type: Logistic ("LOGIT"), linear ("LM"), or Cox proportional hazards ("COX")
rankingTest Variables will be ranked based on: The z-score of the IDI ("zIDI"), the z-score of the NRI ("zNRI"), the IDI ("IDI"), the NRI ("NRI"), the NeRI ("NeRI"), the z-score of the model fit ("Ztest"), the AUC ("AUC"), the Somers' rank correlation ("Cstat"), or the Kendall rank correlation ("Kendall")
cateGroups A vector of percentiles to be used for the categorization procedure
rawDataFrame A data frame similar to data, but with unadjusted data, used to get the means and variances of the unadjusted data
description The name of the column in variableList that stores the variable description
uniType Type of univariate analysis: Binary classification ("Binary") or regression ("Regression")
FullAnalysis If FALSE it will only order the features according to its z-statistics of the linear model
acovariates the list of covariates
timeOutcome the name of the Time to event feature

Details

This function will create valid dummy categorical variables if, and only if, data has been z-standardized. The p-values provided in cateGroups will be converted to its corresponding z-score, which will then be used to create the categories. If non z-standardized data were to be used, the categorization analysis would return wrong results.

Value

A sorted data frame. In the case of a binary classification analysis, the data frame will have the following columns:

Name Name of the raw variable or of the dummy variable if the data has been categorized
parent Name of the raw variable from which the dummy variable was created
univariateRankVariables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>descrip</td>
<td>Description of the parent variable, as defined in description</td>
</tr>
<tr>
<td>cohortMean</td>
<td>Mean value of the variable</td>
</tr>
<tr>
<td>cohortStd</td>
<td>Standard deviation of the variable</td>
</tr>
<tr>
<td>cohortKSD</td>
<td>D statistic of the KS test when comparing a normal distribution and the distribuition of the variable</td>
</tr>
<tr>
<td>cohortKSP</td>
<td>Associated p-value to the cohortKSD</td>
</tr>
<tr>
<td>caseMean</td>
<td>Mean value of cases (subjects with outcome equal to 1)</td>
</tr>
<tr>
<td>caseStd</td>
<td>Standard deviation of cases</td>
</tr>
<tr>
<td>caseKSD</td>
<td>D statistic of the KS test when comparing a normal distribution and the distribution of the variable only for cases</td>
</tr>
<tr>
<td>caseKSP</td>
<td>Associated p-value to the caseKSD</td>
</tr>
<tr>
<td>caseZKSD</td>
<td>D statistic of the KS test when comparing a normal distribution and the distribution of the z-standardized variable only for cases</td>
</tr>
<tr>
<td>caseZKSP</td>
<td>Associated p-value to the caseZKSP</td>
</tr>
<tr>
<td>controlMean</td>
<td>Mean value of controls (subjects with outcome equal to 0)</td>
</tr>
<tr>
<td>controlStd</td>
<td>Standard deviation of controls</td>
</tr>
<tr>
<td>controlKSD</td>
<td>D statistic of the KS test when comparing a normal distribution and the distribution of the variable only for controls</td>
</tr>
<tr>
<td>controlKSP</td>
<td>Associated p-value to the controlsKSD</td>
</tr>
<tr>
<td>controlZKSD</td>
<td>D statistic of the KS test when comparing a normal distribution and the distribution of the z-standardized variable only for controls</td>
</tr>
<tr>
<td>controlZKSP</td>
<td>Associated p-value to the controlsZKSD</td>
</tr>
<tr>
<td>t.Rawvalue</td>
<td>Normal inverse p-value (z-value) of the t-test performed on raw.dataFrame</td>
</tr>
<tr>
<td>t.Zvalue</td>
<td>z-value of the t-test performed on data</td>
</tr>
<tr>
<td>wilcox.Zvalue</td>
<td>z-value of the Wilcoxon rank-sum test performed on data</td>
</tr>
<tr>
<td>ZGLM</td>
<td>z-value returned by the lm, glm, or coxph functions for the z-standardized variable</td>
</tr>
<tr>
<td>zNRI</td>
<td>z-value returned by the improveProb function (Hmisc package) when evaluating the NRI</td>
</tr>
<tr>
<td>zIDI</td>
<td>z-value returned by the improveProb function (Hmisc package) when evaluating the IDI</td>
</tr>
<tr>
<td>zNeRI</td>
<td>z-value returned by the improvedResiduals function when evaluating the NeRI</td>
</tr>
<tr>
<td>ROCAUC</td>
<td>Area under the ROC curve returned by the roc function (pROC package)</td>
</tr>
<tr>
<td>cStatCorr</td>
<td>Somers' rank correlation returned by the rcorr.cens function (Hmisc package)</td>
</tr>
<tr>
<td>NRI</td>
<td>NRI returned by the improveProb function (Hmisc package)</td>
</tr>
<tr>
<td>IDI</td>
<td>IDI returned by the improveProb function (Hmisc package)</td>
</tr>
<tr>
<td>NeRI</td>
<td>NeRI returned by the improvedResiduals function</td>
</tr>
<tr>
<td>kendall.r</td>
<td>Kendall τ rank correlation coefficient between the variable and the binary outcome</td>
</tr>
</tbody>
</table>
In the case of regression analysis, the data frame will have the following columns:

- **Name**: Name of the raw variable or of the dummy variable if the data has been categorized
- **parent**: Name of the raw variable from which the dummy variable was created
- **descrip**: Description of the parent variable, as defined in description
- **cohortMean**: Mean value of the variable
- **cohortStd**: Standard deviation of the variable
- **cohortKSD**: D statistic of the KS test when comparing a normal distribution and the distribution of the variable
- **cohortKSP**: Associated p-value to the cohortKSP
- **cohortZKSD**: D statistic of the KS test when comparing a normal distribution and the distribution of the z-standardized variable
- **cohortZKSP**: Associated p-value to the cohortZKSD
- **ZGLM**: z-value returned by the glm or Cox procedure for the z-standardized variable
- **zNRI**: z-value returned by the improveProb function (`Hmisc` package) when evaluating the NRI
- **NeRI**: NeRI returned by the improvedResiduals function
- **cStatCorr**: c index of Somers’ rank correlation returned by the `rcorr.cens` function (`Hmisc` package)
- **spearman.r**: Spearman \( \rho \) rank correlation coefficient between the variable and the outcome
- **pearson.r**: Pearson \( r \) product-moment correlation coefficient between the variable and the outcome
- **kendall.r**: Kendall \( \tau \) rank correlation coefficient between the variable and the outcome
- **kendall.p**: Associated p-value to the kendall.r
- **TstudentRes.p**: \( p \)-value of the improvement in residuals, as evaluated by the paired t-test
- **WilcoxonRes.p**: \( p \)-value of the improvement in residuals, as evaluated by the paired Wilcoxon rank-sum test
- **FRes.p**: \( p \)-value of the improvement in residual variance, as evaluated by the F-test
update.uniRankVar

Author(s)

Jose G. Tamez-Pena

References


update.uniRankVar

*Update the univariate analysis using new data*

Description

This function updates the results from an univariate analysis using a new data set.

Usage

```r
## S3 method for class 'uniRankVar'
update(object, 
       ...)
```

Arguments

- `object`: A list with the results from the `uniRankVar` function
- `...`: Additional parameters to be passed to the `uniRankVar` function, used to update the univariate analysis

Value

A list with the same format as the one yielded by the `uniRankVar` function

Author(s)

Jose G. Tamez-Pena

See Also

`uniRankVar`
updateModel.Bin  

Update the IDI/NRI-based model using new data or new threshold values

Description

This function will take the frequency-ranked set of variables and will generate a new model with terms that meet either the integrated discrimination improvement (IDI), or the net reclassification improvement (NRI), threshold criteria.

Usage

```r
updateModel.Bin(Outcome, 
  covariates = "1", 
  pvalue = c(0.025, 0.05), 
  VarFrequencyTable, 
  variableList, 
  data, 
  type = c("LM", "LOGIT", "COX"), 
  lastTopVariable = 0, 
  timeOutcome = "Time", 
  selectionType = c("zIDI", "zNRI"), 
  maxTrainModelSize = 0, 
  zthrs = NULL 
)
```

Arguments

- **Outcome**: The name of the column in `data` that stores the variable to be predicted by the model
- **covariates**: A string of the type "1 + var1 + var2" that defines which variables will always be included in the models (as covariates)
- **pvalue**: The maximum $p$-value, associated to either IDI or NRI, allowed for a term in the model
- **VarFrequencyTable**: An array with the ranked frequencies of the features, (e.g. the ranked.var.value returned by the `ForwardSelection.Model.Bin` function)
- **variableList**: A data frame with two columns. The first one must have the names of the candidate variables and the other one the description of such variables
- **data**: A data frame where all variables are stored in different columns
- **type**: Fit type: Logistic ("LOGIT"), linear ("LM"), or Cox proportional hazards ("COX")
- **lastTopVariable**: The maximum number of variables to be tested
- **timeOutcome**: The name of the column in `data` that stores the time to event (needed only for a Cox proportional hazards regression model fitting)
updateModel.Res

selectionType  The type of index to be evaluated by the improveProb function (Hmisc package): z-score of IDI or of NRI
maxTrainModelSize  Maximum number of terms that can be included in the model
zthrs  The z-thresholds estimated in forward selection

Value
final.model  An object of class lm, glm, or coxph containing the final model
var.names  A vector with the names of the features that were included in the final model
formula  An object of class formula with the formula used to fit the final model
z.selectionType  A vector in which each term represents the z-score of the index defined in selectionType obtained with the Full model and the model without one term

Author(s)
Jose G. Tamez-Pena and Antonio Martinez-Torteya

See Also
updateModel.Res

updateModel.Res  Update the NeRI-based model using new data or new threshold values

Description
This function will take the frequency-ranked set of variables and will generate a new model with terms that meet the net residual improvement (NeRI) threshold criteria.

Usage
updateModel.Res(Outcome,
covariates = "1",
pvalue = c(0.025, 0.05),
VarFrequencyTable,
variableList,
data,
type = c("LM", "LOGIT", "COX"),
testType=c("Binomial", "Wilcox", "tStudent"),
lastTopVariable = 0,
timeOutcome = "Time",
maxTrainModelSize = -1,
p.thresholds = NULL
)
Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Outcome</td>
<td>The name of the column in data that stores the variable to be predicted by the model</td>
</tr>
<tr>
<td>covariates</td>
<td>A string of the type &quot;1 + var1 + var2&quot; that defines which variables will always be included in the models (as covariates)</td>
</tr>
<tr>
<td>pvalue</td>
<td>The maximum p-value, associated to the NeRI, allowed for a term in the model</td>
</tr>
<tr>
<td>VarFrequencyTable</td>
<td>An array with the ranked frequencies of the features, (e.g. the ranked.var value returned by the ForwardSelection.Model.Res function)</td>
</tr>
<tr>
<td>variableList</td>
<td>A data frame with two columns. The first one must have the names of the candidate variables and the other one the description of such variables</td>
</tr>
<tr>
<td>data</td>
<td>A data frame where all variables are stored in different columns</td>
</tr>
<tr>
<td>type</td>
<td>Fit type: Logistic (&quot;LOGIT&quot;), linear (&quot;LM&quot;), or Cox proportional hazards (&quot;COX&quot;)</td>
</tr>
<tr>
<td>testType</td>
<td>Type of non-parametric test to be evaluated by the improvedResiduals function: Binomial test (&quot;Binomial&quot;), Wilcoxon rank-sum test (&quot;Wilcox&quot;), Student’s t-test (&quot;tStudent&quot;), or F-test (&quot;Ftest&quot;)</td>
</tr>
<tr>
<td>lastTopVariable</td>
<td>The maximum number of variables to be tested</td>
</tr>
<tr>
<td>timeOutcome</td>
<td>The name of the column in data that stores the time to event (needed only for a Cox proportional hazards regression model fitting)</td>
</tr>
<tr>
<td>maxTrainModelSize</td>
<td>Maximum number of terms that can be included in the model</td>
</tr>
<tr>
<td>p.thresholds</td>
<td>The p.value thresholds estimated in forward selection</td>
</tr>
</tbody>
</table>

Value

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>final.model</td>
<td>An object of class lm, glm, or coxph containing the final model</td>
</tr>
<tr>
<td>var.names</td>
<td>A vector with the names of the features that were included in the final model</td>
</tr>
<tr>
<td>formula</td>
<td>An object of class formula with the formula used to fit the final model</td>
</tr>
<tr>
<td>z.NeRI</td>
<td>A vector in which each element represents the z-score of the NeRI, associated to the testType, for each feature found in the final model</td>
</tr>
</tbody>
</table>

Author(s)

Jose G. Tamez-Pena and Antonio Martinez-Torteya

See Also

updateModel.Bin
Index

*Topic **Benchmarking**
  - barPlotCiError, 13
  - benchmarking, 14
*Topic **Data_Conditioning**
  - featureAdjustment, 45
  - nearestNeighborImpute, 72
  - rankInverseNormalDataFrame, 85
*Topic **Data_Inspection**
  - heatMaps, 63
  - listTopCorrelatedVariables, 68
  - timeSerieAnalysis, 93
  - uniRankVar, 94
  - univariateRankVariables, 97
  - update.uniRankVar, 101
*Topic **Datasets**
  - cancerVarNames, 31
*Topic **Feature_Filtering**
  - FilterUnivariate, 46
  - mRMR.classic_FRESA, 71
*Topic **Feature_Selection**
  - getSignature, 58
  - signatureDistance, 89
*Topic **Hypothesis_Testing**
  - EmpiricalSurvDifff, 42
*Topic **Model_CV**
  - FRESAScale, 56
  - randomCV, 82
*Topic **Model_Diagnosis**
  - bootstrapValidation_Bin, 18
  - bootstrapValidation_Res, 21
*Topic **Model_Generation**
  - backVarElimination_Bin, 8
  - backVarElimination_Res, 10
  - baggedModel, 11
  - bootstrapVarElimination_Bin, 23
  - bootstrapVarElimination_Res, 25
  - BSWiMS.model, 26
  - crossValidationFeatureSelection_Bin, 32
  - crossValidationFeatureSelection_Res, 37
  - CVsignature, 41
  - ForwardSelection.Model.Bin, 48
  - ForwardSelection.Model.Res, 50
  - FRESA.Model, 52
  - KNN_method, 66
  - LASSO, 67
  - LM_RIDGE_MIN, 69
  - NAIVE_BAYES, 71
  - updateModel.Bin, 102
  - updateModel.Res, 103
*Topic **Model_Inspection**
  - ensemblePredict, 44
  - getKNNpredictionFromFormula, 57
  - getVar.Bin, 60
  - getVar.Res, 61
  - improvedResiduals, 65
  - modelFitting, 70
  - plot.bootstrapValidation_Bin, 73
  - plot.bootstrapValidation_Res, 74
  - plot.FRESA_benchmark, 75
  - plotModels.ROC, 76
  - predictionStats, 81
  - reportEquivalentVariables, 87
  - residualForFRESA, 88
  - summary.bootstrapValidation_Bin, 90
  - summary.fitFRESA, 91
  - summaryReport, 92
*Topic **Model_Prediction**
  - predict.fitFRESA, 77
  - predict.FRESA_LASSO, 79
  - predict.FRESA_NAIVEBAYES, 80
  - predict.FRESA_RIDGE, 80
  - predict.FRESAKNN, 78
  - predict.FRESAsignature, 78
*Topic **package**
  - FRESA.CAD-package, 3
backVarElimination_Bin, 8, 11, 24
backVarElimination_Res, 9, 10, 24, 26
baggedModel, 11
barPlotCIError, 13
benchmarking, 14
BinaryBenchmark, 75
BinaryBenchmark (benchmarking), 14
bootstrapValidation_Bin, 18, 23, 91
bootstrapValidation_Res, 20, 21, 26, 91
bootstrapVarElimination_Bin, 9, 11, 23, 26
bootstrapVarElimination_Res, 9, 11, 24, 25, 41
BSWiMS.model, 26
cancerVarNames, 31
correlated_Remove (FilterUnivariate), 46
crossValidationFeatureSelection_Bin, 32, 41
crossValidationFeatureSelection_Res, 36, 37
CVsignature, 41, 78, 79
EmpiricalSurvDiff, 42
ensemblePredict, 12, 44
featureAdjustment, 45
FilterUnivariate, 46
ForwardSelection.Model.Bin, 36, 48, 52
ForwardSelection.Model.Res, 36, 50, 50
FRESA.CAD (FRESA.CAD-package), 3
FRESA.CAD-package, 3
FRESA.Model, 52
FRESAScale, 56, 67
getKNNpredictionFromFormula, 57
getSignature, 41, 58, 79
getVar.Bin, 60, 63
getVar.Res, 61, 61
heatMaps, 63
improvedResiduals, 41, 65
knn, 58
KNN_method, 66, 78
LASSO, 67, 79
LASSO_1SE (LASSO), 67
LASSO_MIN (LASSO), 67
listTopCorrelatedVariables, 68
LM_RIDGE_MIN, 69, 80, 81
modelFitting, 70, 91
mRMR.classic_FRESA, 71
NAIVE_BAYES, 71, 80
nearestNeighborImpute, 72, 77
OrdinalBenchmark (benchmarking), 14
plot (plot.bootstrapValidation_Bin), 73
plot.bootstrapValidation_Bin, 20, 73, 74
plot.bootstrapValidation_Res, 23, 74, 74
plot.FRESA_benchmark, 75
plotModels.ROC, 76
predict (predict.fitFRESA), 77
predict.fitFRESA, 58, 77
predict.FRESA_LASSO, 79
predict.FRESA NaiveBayes, 80
predict.FRESA Ridge, 80
predict.FRESAKNN, 78
predict.FRESA_Signature, 78
predictionStats, 81
predictionStats_binary, 75
predictionStats_binary (predictionStats), 81
predictionStats_ordinal (predictionStats), 81
predictionStats_regression (predictionStats), 81
randomCV, 14, 16, 82, 82
rankInverseNormalDataFrame, 57, 85
RegressionBenchmark (benchmarking), 14
reportEquivalentVariables, 87
residualForFRESA, 88
signatureDistance, 41, 79, 89
summary (summary.fitFRESA), 91
summary.bootstrapValidation_Bin, 20, 90, 92
summary.fitFRESA, 91
summaryReport, 91, 92
timeSerieAnalysis, 93
uniRankVar, 94, 101
univariate_correlation (FilterUnivariate), 46
univariate_Logit (FilterUnivariate), 46
univariate_residual (FilterUnivariate), 46
univariate_tstudent (FilterUnivariate), 46
univariate_Wilcoxon (FilterUnivariate), 46
univariateRankVariables, 96, 97
update (update.uniRankVar), 101
update.uniRankVar, 96, 101
updateModel.Bin, 102, 104
updateModel.Res, 103, 103